UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

Scuola Politecnica e delle Scienze di Base

Area Didattica di Scienze Matematiche Fisiche e Naturali

Dipartimento di Fisica "Ettore Pancini"

Laurea Magistrale in Fisica

Properties of molecular crystals using machine learning potentials

Relatori
Prof. Dario Alfè
Prof. Andrea Zen

Candidato
Mariano Mollo
Matr. N94000618

Anno Accademico 2023/2024

Abstract

Molecular crystals play an important role in the field of materials science, particularly in drug development, electronics, and renewable energy sectors.

In this work, we will use recently developed Machine Learning Potentials (MLPs) to model the behaviour and characteristics of molecular crystals, using water as the object of study, given the vast coverage in literature.

Traditional approaches often grapple with the trade-off between computational expense and accuracy. The application of MLPs captures complex intermolecular interactions with a significantly reduced computational cost compared to traditional ab-initio methods.

We will study the capabilities of trained MLPs to accurately predict lattice energies, polymorphic behaviours, and response to external conditions like temperature and pressure. We will also study dynamic properties such as phonon spectra to complete the insight into the physical and chemical behaviours of molecular crystals.

Contents

List of Figures

List of Tables

List of Code Blocks

Glossary

***Adaline* – ADAptive LInear Neuron**. 20

***ASE* – Atomic Simulation Environment**. 38, 40, 49, 65, 66, 71, 73

***CBS* – complete basis set**. 46

***CCSD(T)* – coupled cluster theory with singlets, doublets, and perturbative triplets**. 46

***CNN* – Convolutional Neural Network**. 32

***CSP* – Crystal Structure Prediction**.

***DFT* – Density Functional Theory**. vii, 1, 12, 18, 19, 35, 49, 50, 71

***DMC* – Diffusion Monte Carlo**. 51, 54

***GGA* – Generalized Gradient Approximation**. 19

***GNN* – Graph Neural Network.** 28, 29, 33, 34, 67, 68

***HDNNP* – high-dimensional neural network potential.** 2

***IBiSCo* – Infrastructure for BIg data and Scientific Computing.** 65

***KS* – Kohn-Sham.** 13

***MAE* – Mean Absolute Error.** ix, 48, 53

***MD* – Molecular Dynamics.** 14, 15, 16, 62

***ML* – Machine Learning.** 70

***MLP* – Machine Learning Potential.** iii, 2, 3, 35, 53, 71, 75

***MPNN* – Message Passing Neural Network.** 29, 67, 68

***NN* – Neural Network.** 20, 22

***PAW* – Projector-Augmented plane Wave.** 71

***PBE* – Perdew-Burke-Ernzerhof.** 71

***PES* – Potential Energy Surface.**

***PI* – Path Integral.**

***QTI* – Quantum Thermodynamic Integration.**

***RDF* – Radial Distribution Function.** 62

***T&QN* – Thermal and Quantum Nuclear.**

***VASP* – Vienna Ab initio Simulation Package.** 71

***vdW* – van der Waals.**

***XC* – Exchange–Correlation.** 53

***XDM* – Exchange-hole Dipole Moment.** 18, 19

***ZPE* – Zero Point Energy.** 18, 42, 43

1 INTRODUCTION

Molecular crystals represent a significant area of study within materials science due to their diverse applications in fields such as pharmaceuticals, electronics, and renewable energy. In drug development, for instance, the properties of molecular crystals directly influence the effectiveness and delivery of medications, as exemplified in Figure 1. [21] Beyond pharmaceuticals, these materials are key to advancing technologies involving semiconductors and energy storage systems, given their tunable electronic and optical properties. [22], [23], [1], [24]

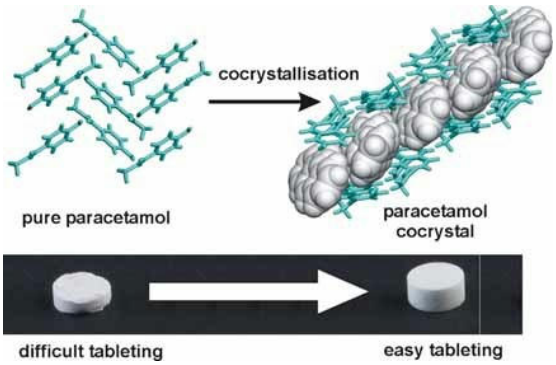


Figure 1: Poor mechanical properties of paracetamol are improved through the strategy of cocrystal formation. Graphical Abstract taken from [1].

Despite the critical importance [25] of understanding and predicting the properties of molecular crystals, traditional computational approaches face challenges in balancing accuracy with computational expense. Methods such as DFT offer accurate insights into molecular interactions but are prohibitively expensive for large-scale simulations. Conversely, classical force field methods scale well but often lack the precision needed for complex systems. This trade-off has driven the exploration of alternative methods that combine the best of both worlds.

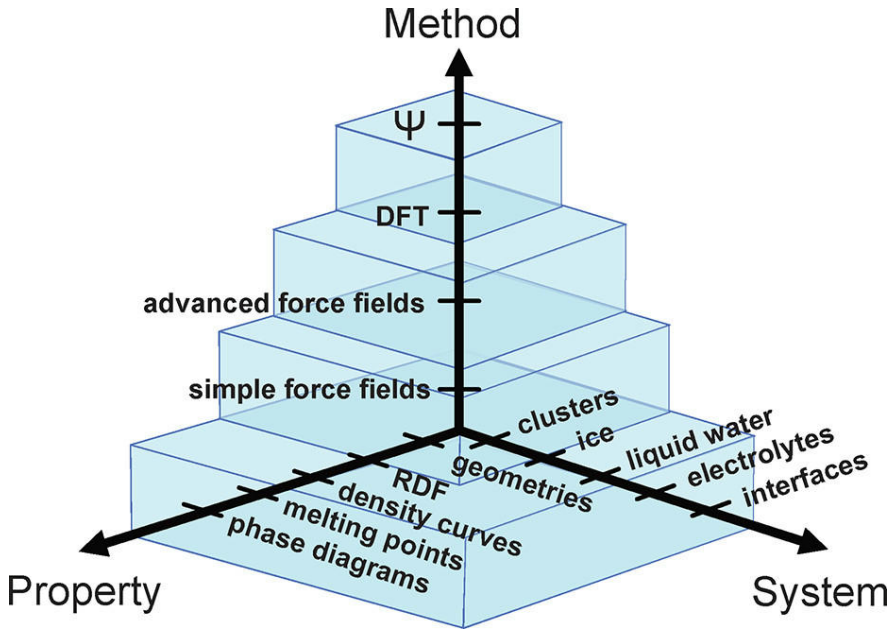


Figure 2: From [2, Figure 1]. Pyramid of potentials for atomistic simulations illustrated for the example of water and aqueous systems. Using high-level wave function-based methods as represented by Ψ only the geometries of small systems such as water clusters in vacuum are accessible, while DFT is the standard method to determine simple properties such as RDFs of liquid water in ab initio molecular dynamics simulations. Very large-scale simulations of complex systems such as electrolytes or solid−liquid interfaces, or the determination of complex thermodynamic properties, can only be carried out using atomistic potentials such as forcefields. Also MLPs allow the study of these systems.

Recently, MLPs have emerges as a promising solution. These approaches leverage advanced neural network architectures to model the potential energy surfaces of molecular systems with a level of accuracy comparable to ab initio methods but at a fraction of the computational cost. By training on existing quantum mechanical data, MLPs can predict intermolecular interactions and dynamic behaviours with high fidelity, making them suitable for studying complex systems like molecular crystals. [26]–[29]

Since their introduction about 25 years ago, machine learning (ML) potentials have become an important tool in the field of atomistic simulations. After the initial decade, in which neural networks were successfully used to construct potentials for rather small molecular systems, the development of high-dimensional neural network potentials (HDNNPs) in 2007 opened the way for the application of ML potentials in simulations of large systems containing thousands of atoms. [2]

It has been estimated that mixed ab-initio and MLP calculations require between three and ten times less the core hours of purely physical functionals. [29]

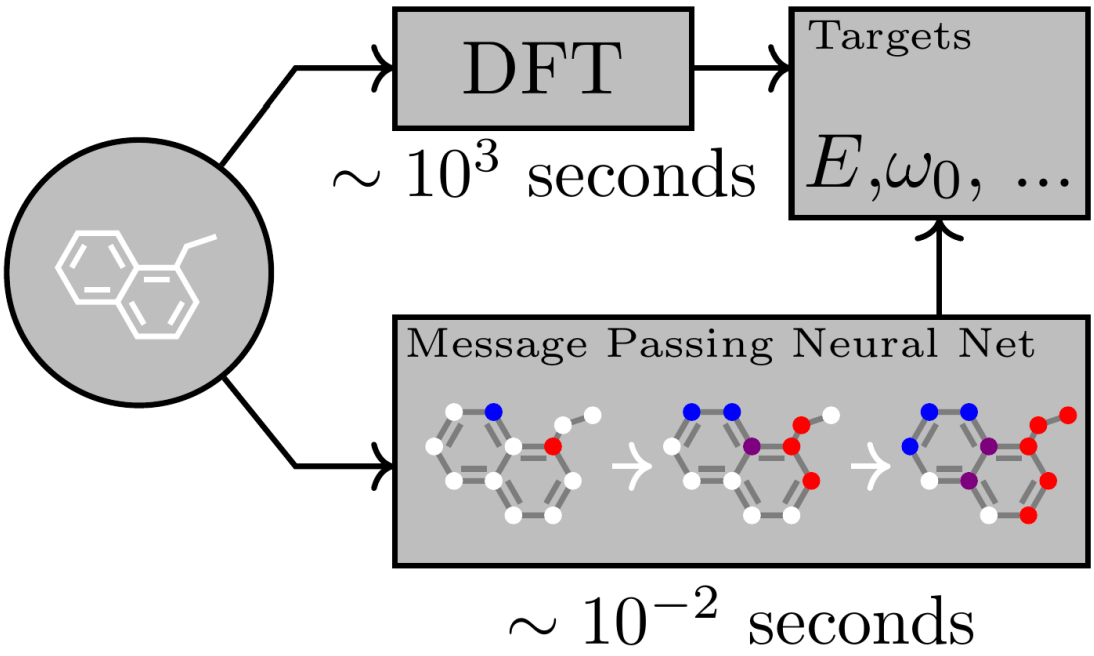


Figure 3: A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation. Image taken from [3].

This thesis investigates the capabilities of MLPs, with a specific focus on modeling the properties of water as a molecular crystal. Water serves as an ideal test case due to its well-documented polymorphic behaviours and the wealth of literature available for benchmarking. [27], [30] The research primarily explores the accuracy of MLPs in predicting molecular structural stabilities, lattice energies of different ice polymorphs, and dynamical properties of crystals, such as phonon spectra.

The following chapters will introduce the theoretical foundations and the tools needed to pursue this question. Section 2 briefly describes the theoretical foundation underlying the physical priciples and the algorithms on which MLPs like MACE are built onto. Section 3 and Section 4 show the results obtained through the test of the new MACE calculator on known atomic configurations and discuss its performances. Section 5 details the tools employed to run the computer simulation experiments; the algorithms section, Section 5.3, mirrors and details the practical implementation of the objects first outlined in Section 2.6.

2 | THEORY

2.1 THREE DIMENSIONAL HARMONIC CRYSTALLINE SOLIDS

The crystal can be described by a collection of independent harmonic oscillators with potential energy:

$$U = U_0 + \sum_{i=1}^{3N} \frac{1}{2} M \omega_i^2 q_i^2, \tag{1}$$

where M is the mass of the particles, q_i a set of *normal coordinates*, and U_0 the energy of the system in its ground state. This approximation is known as the *harmonic approximation*. The Newton's equations of motion for the normal coordinates are:

$$M \frac{\mathrm{d}^2 q_i}{\mathrm{d}t^2} = -\frac{\partial U}{\partial q_i} - M\omega_i^2 q_i. \tag{2}$$

Consider a three dimensional system made of particles arranged on a *periodic lattice*, at equilibrium positions $\{\vec{r}^{\,0}\} := \vec{r}_1^{\,0}, ..., \vec{r}_N^{\,0}, ...$ This *Bravais lattice* is completely defined by a set of three *lattice vectors*, $\vec{a}_1, \vec{a}_2, \vec{a}_3$, and every point on the lattice can be obtained as:

$$\vec{r}_j^{\,0} = n\vec{a}_1 + m\vec{a}_2 + l\vec{a}_3, \tag{3}$$

where j is a shorthand for the three integers n, m, l. We also define the *reciprocal lattice vectors*:

$$\vec{b}_1 := 2\pi \frac{\vec{a}_2 \times \vec{a}_3}{V}, \quad \vec{b}_2 := 2\pi \frac{\vec{a}_3 \times \vec{a}_1}{V}, \quad \vec{b}_3 := 2\pi \frac{\vec{a}_1 \times \vec{a}_2}{V}, \tag{4}$$

where $V := (\vec{a}_1 \times \vec{a}_2) \cdot \vec{a}_3$ is the volume of the *primitive cell*. This is the smallest unit of volume that can be periodically repeated to fill the whole space, and clearly it can be constructed in multiple (infinite) ways. A convenient construction is the *Wigner-Seitz* primitive cell, which is the locus of points in space closer to a particular lattice point than to any other lattice point; this procedure is the application of Voronoi decomposition to a crystal lattice. The reciprocal lattice vectors span the *reciprocal space*, which is also a periodic lattice with points at positions:

$$\vec{g} := n'\vec{b}_1 + m'\vec{b}_2 + l'\vec{b}_3, \tag{5}$$

with n', m', l' integer numbers. The Wigner-Seitz cell in reciprocal space is called the *first Brillouin zone*. The reciprocal vectors and the lattice vectors satisfy the orthogonality relations:

$$\vec{b}_i \cdot \vec{a}_j = 2\pi\delta_{ij}. \tag{6}$$

Each lattice site does not need to be occupied by just one particle; indeed, the generalization to crystals with more than one particle per lattice site is straightforward.

Let $U_{\{\vec{r}^0\}}(\{\vec{r}\})$ be the potential energy function, defined by the interacting particles when they are in their equilibrium positions $\{\vec{r}^0\}$. With this we mean that if we displace the particles by amounts $\vec{u}_i := \vec{r}_i - \vec{r}_i^0$, we assume that these displacements are not changing the functional form of the potential, but only the computed value changes. This assumption can only be valid if the displacements $\{\vec{u}\}$ are small. An extreme case where this cannot possibly hold is when the atoms move so much that the crystal assumes a different crystal structure, or it melts. For zero displacements, when all particles are in their equilibrium positions, the potential energy can be computed from knowledge of the lattice vectors only, i.e., one only needs information about the primitive cell. If instead the particles are displaced from their equilibrium positions, then we need the positions of all of them. For small enough displacements $\{\vec{u}\}$ the potential energy can be expanded around its minimum, where all particles are in their equilibrium positions $\{\vec{r}^0\}$:

$$U(\{\vec{r}\}) = U_0 + \frac{1}{2}\sum_{i,j} \vec{u}_i \cdot \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \vec{u}_j + \cancel{\Theta}(u^3), \tag{7}$$

where $U_0 := U(\{\vec{r}^0\})$, and the linear term is absent because we are expanding around the minimum of the potential. The *force constants matrix* Φ is defined as:

$$\Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) := \left(\frac{\partial^2 U(\{\vec{r}\})}{\partial\vec{r}_i \partial\vec{r}_j}\right)_{\substack{\vec{r}_i=\vec{r}_i^0 \\ \vec{r}_j=\vec{r}_j^0}} = \begin{pmatrix} \varphi_{ij}^{xx} & \varphi_{ij}^{xy} & \varphi_{ij}^{xz} \\ \varphi_{ij}^{yx} & \varphi_{ij}^{yy} & \varphi_{ij}^{yz} \\ \varphi_{ij}^{zx} & \varphi_{ij}^{zy} & \varphi_{ij}^{zz} \end{pmatrix} \tag{8}$$

$$\varphi_{ij}^{\alpha\beta} = \frac{\partial^2 U(\{\vec{r}\})}{\partial\alpha_i \partial\beta_i},$$

where α_i and β_j run over the three cartesian components of $\vec{r}_i^0$ and $\vec{r}_j^0$. If the displacements are not too large and the $\cancel{\Theta}(u^3)$ terms can be ignored, the force

acting on particle at position $\vec{r}_i^0$ due to the displacements $\vec{u}_j$ of all particles in the system, including $\vec{u}_i$, is:

$$\vec{f}_i = -\sum_j \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \vec{u}_j. \tag{9}$$

The dependence of the force constants matrix on the difference $\vec{r}_i^0 - \vec{r}_j^0$ rather than on $\vec{r}_i^0$ and $\vec{r}_j^0$ separately is due to translational invariance. The force constants matrix satisfies other important properties:

1. $\Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) = \Phi\left(\vec{r}_j^0 - \vec{r}_i^0\right)$, which is implied by the fact that the double derivative in Equation 8 is invariant upon changing the order of differentiation.
2. $\sum_j \Phi\left(\vec{r}_j^0\right) = 0$. This property can be understood by imagining to displace the whole crystal rigidly by some vector $\vec{c}$. Clearly, such a displacement cannot affect the forces acting on the particles, because the relative differences between them are unaffected by the translation, and so we must have $\vec{f}_i = -\sum_j \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \vec{u}_j = -\sum_j \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \left(\vec{u}_j + \vec{c}\right)$. Therefore, $\sum_j \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \vec{c} = 0$. Since this is independent on the particular choice of $\vec{r}_i^0$ and $\vec{c}$, we must have $\sum_j \Phi\left(\vec{r}_j^0\right) = 0$.

We can obtain the normal modes of the system and show that the potential energy is given by the sum of squares of the normal modes. To obtain the dispersion relation, consider the Newton's equation of motion for a particle at one particular lattice position, $\vec{r}_i^0$:

$$M\frac{\mathrm{d}^2\vec{u}_i}{\mathrm{d}t^2} = \vec{f}_i. \tag{10}$$

The motion of the particle around its equilibrium position can be described by the form:

$$\vec{u}_i(t) \propto \vec{\varepsilon} e^{i\left(\vec{q}\cdot\vec{r}_i^0 - \omega t\right)}, \tag{11}$$

where $\vec{\varepsilon}$ is a *polarization vector*, which defines the direction of oscillation of the particle, and $\vec{q}$ is a *wave vector*. The physical motion of the particles is obtained by taking the real part of Equation 11. Substituting Equation 11 into Equation 10 we obtain:

$$M\omega^2\vec{\varepsilon} = \sum_j e^{i\vec{q}\cdot\left(\vec{r}_j^0 - \vec{r}_i^0\right)} \Phi\left(\vec{r}_i^0 - \vec{r}_j^0\right) \cdot \vec{\varepsilon}. \tag{12}$$

The sum over j runs over all lattice sites, and so it is independent on our choice of $\vec{r}_i^0$. We can therefore choose $\vec{r}_i^0 = \vec{0}$ and replace the difference $\vec{r}_j^0 - \vec{r}_i^0$ simply with $\vec{r}_j^0$.

2.1.1 Phonons

If we introduce the *dynamical matrix*:

$$D(\vec{q}) := \frac{1}{M} \sum_j e^{i\vec{q}\cdot\vec{r}_j^0} \Phi\left(\vec{r}_j^0\right) \tag{13}$$

we can rewrite Equation 12 as:

$$\omega^2 \vec{\varepsilon} = D(\vec{q}) \cdot \vec{\varepsilon} \tag{14}$$

The transformation in Equation 13 is also known as a *lattice Fourier transform*. Note that it is sufficient to define the dynamical matrix in the first Brillouin zone, because any vector in reciprocal space can be written as the sum of a vector in the Brillouin zone plus a reciprocal lattice vector. We have:

$$e^{i(\vec{q}+\vec{g})\cdot\vec{r}_j^0} = e^{i\vec{q}\cdot\vec{r}_j^0} e^{i\vec{g}\cdot\vec{r}_j^0} = e^{i\vec{q}\cdot\vec{r}_j^0}, \tag{15}$$

because:

$$\vec{g}\cdot\vec{r}_j^0 = (nn' + mm' + ll')2\pi, \tag{16}$$

and so the dynamical matrix has the same periodicity of the reciprocal lattice.

The property $\sum_j \Phi\left(\vec{r}_j^0\right) = 0$ implies $D(\vec{q}) = \vec{0}$; therefore, the dispersion relation gives zero frequencies at zero wavevector. This *sum rule* is very useful as a practical sanity check of the consistency of the calculations.

To solve Equation 14 we need to solve an eigenvalue problem. We need to find a reference frame in which the matrix $D(\vec{q})$ is diagonal. The elements of the diagonal, $\omega^2_{\vec{q},s}$, with $s \in \{1,2,3\}$ representing the *branch number*, are the *eigenvalues*, and the three vectors $\vec{\varepsilon}_{\vec{q},s}$, the polarizations of the normal modes, are the *eigenvectors* that define the reference frame.

Each normal mode represents a collective oscillation of the particles in the system with frequency $\omega_{\vec{q},s}$. These collective oscillations are called *phonons*.

The potential energy can be written in the form in Equation 1 and so the partition function has the form of either Equation 17 or Equation 18, depending on if the system is treated classically or quantum-mechanically:

$$Z = e^{-\beta U_0} \prod_{i=1}^{3N} \frac{k_B T}{\hbar\omega_i} \tag{17}$$

$$
\begin{aligned}
Z &= e^{-\beta U_0} \prod_{i=1}^{3N} Z_i \\
Z_i &= \sum_{n=0}^{\infty} e^{-\frac{E_n^i}{k_B T}} = \sum_{n=0}^{\infty} e^{-(n+\frac{1}{2})\frac{\hbar\omega_i}{k_B T}} = \frac{e^{-\frac{1}{2}\frac{\hbar\omega_i}{k_B T}}}{1 - e^{-\frac{\hbar\omega_i}{k_B T}}}
\end{aligned}
\tag{18}
$$

2.1.2 The Helmholtz energy

The harmonic contribution of phonons to the Helmholtz energy is represented by the equation:

$$
\begin{aligned}
&F_{\text{harm}}(V,T) = \\
&= \frac{1}{\Omega}\int_{\Omega}\sum_{s=1}^{3}\frac{\hbar\omega_{\vec{q},s}}{2}\,\mathrm{d}q + k_B T\frac{1}{\Omega}\int_{\Omega}\sum_{s=1}^{3}\ln\left(1-\exp\left(-\frac{\hbar\omega_{\vec{q},s}}{k_B T}\right)\right)\mathrm{d}q
\end{aligned}
\tag{19}
$$

In the high temperature limit, defined by $\frac{\hbar\omega_{\vec{\omega},s}}{k_B T} \ll 1$, the harmonic term reduces to the classical expression:

$$
F^c_{\text{harm}} = \frac{k_B T}{\Omega}\int_{\Omega}\sum_{s=1}^{3}\ln\left(\frac{\hbar\omega_{\vec{q},s}}{k_B T}\right)\mathrm{d}q \tag{20}
$$

For computation efficiency purposes, the integrals above are approximated with a sum running on points sampled in the Brillouin zone, denoted by BZ in the sum:

$$
\frac{1}{\Omega}\int \mathrm{d}q \approx \frac{1}{N_{\vec{q}}}\sum_{\vec{q}\in\,\text{BZ}} \tag{21}
$$

$$
\begin{aligned}
F(V,T) &= U_0(V) \\
&+ \sum_{s=1}^{3}\sum_{\vec{q}}\left[\frac{\hbar\omega_{\vec{q},s}(V)}{2} + k_B T\ln\left(1-\exp\left(-\frac{\hbar\omega_{\vec{q},s}(V)}{k_B T}\right)\right)\right]
\end{aligned}
\tag{22}
$$

$$
F_{\text{classical}}(V,T) = U_0(V) + k_B T\sum_{s=1}^{3}\sum_{\vec{q}}\ln\frac{\hbar\omega_{\vec{q},s}(V)}{k_B T} \tag{23}
$$

The sum over $\vec{q}$ runs over all vectors in the Brillouin zone, which are infinite for a Bravais lattice that contains an infinite number of sites. Indeed, the Helmholtz energy of a crystal with an infinite number of particles would also be infinite. The physical quantity of interest is therefore the Helmholtz energy per particle; in practice, this is obtained by computing the sums Equation 22 and Equation 23 using a finite grid of $N_{\vec{q}}$ points in the Brillouin zone and dividing the total Helmholtz energy by $N_{\vec{q}}$. Both the ground state energy and the frequen-

cies explicitly depend on the volume V. The relationship linking the frequencies with volume and temperature is responsible for the different temperature dependence of the Helmholtz energy at different volumes; these terms cause the phenomenon of thermal expansion in solids.

2.1.3 The small displacement method

Let us consider the crystal in the ground state and displace one particle from its equilibrium position. Let the displacement be along the x axis, $\vec{u} := (u, 0, 0)$. The forces acting on all particles in the system, including the displaced one, are given by:

$$(f_i^x, f_i^y, f_i^z) = -\Phi(\vec{r}_i^0) \cdot (u, 0, 0) = -u \begin{pmatrix} \varphi_{0i}^{xx} \\ \varphi_{0i}^{yx} \\ \varphi_{0i}^{zx} \end{pmatrix} \tag{24}$$

from which we obtain the first columnt of the force constant matrix:

$$\varphi_{0i}^{xx} = -\frac{f_i^x}{u}, \quad \varphi_{0i}^{yx} = -\frac{f_i^y}{u}, \quad \varphi_{0i}^{zx} = -\frac{f_i^z}{u} \tag{25}$$

To obtain the other columns, we need to repeat the procedure with displacements $(0, u, 0)$ and $(0, 0, u)$. A set of three forces calculations is sufficient to compute the full force constants matrix.

In practical calculations, it is not possible to include all the infinite particles of a lattice; therefore, we use a procedure that approximates the evaluation of the force constants matrix, with an error that can be systematically reduced. We consider a *supercell*, which is an integer multiple of the primitive cell in the three spatial directions. We can define three multiplicative integers, one for each direction, i, j, k. Named $\vec{a}_1, \vec{a}_2, \vec{a}_3$ the lattice vectors of the primitive cell, the periodicity of the supercell is described by the multiplied lattice vectors:

$$\vec{A}_1 := i\vec{a}_1, \quad \vec{A}_2 := j\vec{a}_2, \quad \vec{A}_3 := k\vec{a}_3 \tag{26}$$

The reciprocal lattice vectors of the supercell are defined as follow:

$$\vec{B}_1 := 2\pi \frac{\vec{A}_2 \times \vec{A}_3}{V}, \quad \vec{B}_2 := 2\pi \frac{\vec{A}_3 \times \vec{A}_1}{V}, \quad \vec{B}_3 := 2\pi \frac{\vec{A}_1 \times \vec{A}_2}{V}, \tag{27}$$

with $V := \left(\vec{A}_1 \times \vec{A}_2\right) \cdot \vec{A}_3$ as the volume of the supercell. Following the periodicity of the finite supercell, the displacement by an amount $\vec{u}$ of one particle causes the displacement of all its periodic images too, located at positions $\vec{L} := n\vec{A}_1 + m\vec{A}_2 + l\vec{A}_3$, with n, m, l any three integers. The obtained forces are:

$$\vec{f}_i = -\sum_{\vec{L}} \Phi\left(\vec{r}_i^0 + \vec{L}\right) \cdot \vec{u} \tag{28}$$

where the sum runs over all possible $\vec{L}$ vectors distancing different supercells. We do not have direct access to the elements of $\Phi(\vec{r}_i^0)$, but only to the force constants matrix of the supercell, defined as:

$$\Phi_{\mathrm{SC}}(\vec{r}_i^0) \coloneqq \sum_{\vec{L}} \Phi\left(\vec{r}_i^0 + \vec{L}\right). \tag{29}$$

This also implies that it is impossible to obtain the exact dynamical matrix, which requires knowledge of every element of the force constants matrix. We can rewrite Equation 13 as:

$$\begin{aligned} D(\vec{q}) &= \frac{1}{m} \sum_j \sum_{\vec{L}} e^{i\vec{q}\cdot\left(\vec{r}_j^0 + \vec{L}\right)} \Phi\left(\vec{r}_j^0 + \vec{L}\right) \\ &= \frac{1}{m} \sum_j e^{i\vec{q}\cdot\vec{r}_j^0} \sum_{\vec{L}} e^{i\vec{q}\cdot\vec{L}} \Phi\left(\vec{r}_j^0 + \vec{L}\right) \end{aligned} \tag{30}$$

with the sum over j running on the lattice sites $\vec{r}_j^0$ belonging to the supercell. Consider now the term $\sum_{\vec{L}} e^{i\vec{q}\cdot\vec{L}} \Phi\left(\vec{r}_j^0 + \vec{L}\right)$ and compare it with Equation 29. For every $\vec{q}$ such that $e^{i\vec{q}\cdot\vec{L}} = 1$ we obtain:

$$D(\vec{q}) = \frac{1}{m} \sum_j e^{i\vec{q}\cdot\vec{r}_j^0} \Phi_{\mathrm{SC}}\left(\vec{r}_j^0\right) \tag{31}$$

Since Φ_{SC} can be calculated, at these particular $\vec{q}$ vectors the dynamical matrix is exact. We can carachterize these special $\vec{q}$ vectors as the linear combinations of the reciprocal vectors of the supercell with integer coefficients n', m', l'. We can verify this statement calculating, given $\vec{q} \coloneqq n'\vec{B}_1 + m'\vec{B}_2 + l'\vec{B}_3$ and $\vec{L} \coloneqq n\vec{A}_1 + m\vec{A}_2 + l\vec{A}_3$, $\vec{q} \cdot \vec{L} = (nn' + mm' + ll')2\pi$, which yields $e^{i\vec{q}+\vec{L}} = 1$.

Increasing the size of the supercell, the Brillouin zone is populated with more exact phonons; inbetween those points we obtain a *Fourier interpolation*, which becomes more and more accurate as we increase the size of the supercell. Eventually, the supercell is so large that the force constants matrix is negligible at its edges; when this happens, the only term contributing in the sum in Equation 29 is that with $\vec{L} = \vec{0}$; as we increase the size of the supercell, the supercell force constants matrix asymptotically approaches the force constants matrix, $\Phi_{\mathrm{SC}}(\vec{r}_i^0) \simeq \Phi(\vec{r}_i^0)$. In this limit, the Fourier interpolation is accurate everywhere.

2.2 DFT

Density Functional Theory (DFT) is a formulation of quantum mechanics, used to compute energies and forces, structural and electronic properties of materials in solid-state physics. It is the underlying method used by VASP, employed in this thesis during the fine-tuning of MACE models, described in Section 5.3.5.

Given the hamiltonian of a time-independent many-body Schrödinger equation for a system of N electrons, $\hat{H} = \hat{T} + \hat{V}_{\text{ee}} + \hat{V}_{\text{ext}}$, and the electron density:

$$\rho(\vec{r}) = N \int_V \mathrm{d}^3\vec{r}_2 ... \mathrm{d}^3\vec{r}_N \ |\Psi(\vec{r}, \vec{r}_2, ..., \vec{r}_N)|^2; \tag{32}$$

the potential energy due to the external potential is a *functional* of the electron density, which we indicate with the notation $V_{\text{ext}}[\rho]$:

$$V_{\text{ext}}[\rho] = N \int_V \mathrm{d}^3\vec{r} \rho(\vec{r}) v(\vec{r}). \tag{33}$$

The Hohenberg-Kohn theorems [31] state that $V_{\text{ext}}[\rho_0]$, where ρ_0 is the ground state density, is a *unique* functional of ρ_0. ρ_0 determines Ψ_0, the ground state wavefunction of the system, and thus all the physical properties of the system; we say they are *functionals* of the ground state density. In particular, the ground state energy, E_0, is a functional of ρ_0, as well as the kinetic and electron-electron interaction contributions:

$$E_0 = E[\rho_0] = T[\rho_0] + V_{\text{ee}}[\rho_0] + V_{\text{ext}}[\rho_0]. \tag{34}$$

The Kohn-Sham method [32] provides a working procedure to find the ground state density, defining the Kohn-Sham potential, $v_{\text{KS}}[\rho](\vec{r}) \coloneqq v(\vec{r}) + \int_V \frac{\rho(\vec{r}')}{|\vec{r}-\vec{r}'|}\mathrm{d}^3\vec{r}' + \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho}$, that leads to the Kohn-Sham self-consistent equations:

$$\hat{h}_{\text{KS}}[\rho](\vec{r})\psi_n(\vec{r}) = \left[-\frac{1}{2}\nabla^2 + v_{\text{KS}}[\rho](\vec{r})\right]\psi_n(\vec{r}) = \varepsilon_n\psi_n(\vec{r}) \tag{35}$$

The equations are coupled through the effective potential $v_{\text{KS}}[\rho]$, as $\rho(\vec{r})$ depends on all the $\psi_n(\vec{r})$.

Since the orbitals ψ_n are ortho-normal, the ground state electron density of the system is obtained from the solution of Equations 35 as:

$$\rho_0(\vec{r}) = \sum_{n=1}^{N} |\psi_n(\vec{r})|^2. \tag{36}$$

The extremes of the functional are obtained for any ensemble of N eigenstates of $\hat{h}_{\text{KS}}[\rho]$, and the ground state is obtained by finding the lowest N eigenstates

(or the lowest $\frac{N}{2}$ eigenstates, taking into account the spin degeneracy). One can introduce a set of L basis functions, $\{\varphi_m\}_{m=1,\dots,L}$, to construct the matrix $\varepsilon_{ij} = \left\langle \varphi_i \middle| \hat{h}_{\mathrm{KS}}[\rho] \middle| \varphi_j \right\rangle$, and diagonalize it. The eigenvectors are of the type:

$$\psi_n = \sum_{i=1}^{L} c_i^n \varphi_i \tag{37}$$

If the basis set $\{\varphi_m\}$ is complete, the solution is exact. However, usually this means including an infinite number of elements, which cannot be done in practice; therefore, the solution to the Kohn-Sham (KS) equations is only approximate. By increasing the number of basis functions, one can drive the calculations to convergence, where the energy and other properties are obtained within some predefined threshold. The usual variational principle applies, and so, by including more and more elements in the basis set, the ground state energy decreases monotonically. The rate of decrease of the energy can be used to judge the level of convergence.

Since the KS potential depends on ρ, Equations 35 have to be solved self-consistently. This is typically done by iteration, in which one starts with some initial guess for the electron density ρ_1, constructs $v_{\mathrm{KS}}[\rho_1]$ and solves Equations 35. With those solutions construct ρ_2 using Equation 36 or more advanced algorithms, and solve Equations 35 again, using the newly constructed $v_{\mathrm{KS}}[\rho_2]$. The algorithm runs until the difference between ρ_{j+1} and ρ_j is below some acceptable threshold.

Multiplying Equations 35 by $\psi_n^\star(\vec{r})$, integrating over $\vec{r}$, summing over n, the total energy can be obtained from:

$$E = 2\sum_{n=1}^{\frac{N}{2}} \varepsilon_n - \frac{1}{2}\int_V \frac{\rho(\vec{r}\,')\rho(\vec{r})}{|\vec{r}-\vec{r}\,'|}\mathrm{d}^3\vec{r}\,'\mathrm{d}^3\vec{r} - \int_V \frac{\delta E_{\mathrm{xc}}}{\delta\rho(\vec{r})}\rho(\vec{r})\mathrm{d}^3\vec{r} + E_{\mathrm{xc}}. \tag{38}$$

2.3 MOLECULAR DYNAMICS

2.3.1 The Verlet algorithm

The Verlet algorithm is a technique to generate the trajectory of interacting particles obeying the Newton's equations of motion. [33] It is a discretization of Newton's equations of motion:

$$\vec{f}_i = M\ddot{\vec{r}}_i = -\frac{\partial U(\{\vec{r}\})}{\partial \vec{r}_i} \tag{39}$$

Let us consider the Taylor expansion of the position of particle i at time t, $\vec{r}_i(t)$, computed with forward and backward differences:

$$\vec{r}_i(t+\delta t) = \vec{r}_i(t) + \dot{\vec{r}}_i(t)\delta t + \frac{1}{2}\ddot{\vec{r}}_i(t)(\delta t)^2 + \frac{1}{3!}\dddot{\vec{r}}_i(t)(\delta t)^3 + O\Big((\delta t)^4\Big) \quad (40)$$

$$\vec{r}_i(t-\delta t) = \vec{r}_i(t) - \dot{\vec{r}}_i(t)\delta t + \frac{1}{2}\ddot{\vec{r}}_i(t)(\delta t)^2 - \frac{1}{3!}\dddot{\vec{r}}_i(t)(\delta t)^3 + O\Big((\delta t)^4\Big) \quad (41)$$

Summing the two equations side by side, we obtain:

$$\vec{r}_i(t+\delta t) + \vec{r}_i(t-\delta t) = 2\vec{r}_i + \ddot{\vec{r}}_i(t)(\delta t)^2 + O\Big((\delta t)^4\Big) \quad (42)$$

Consider the expression of $\ddot{\vec{r}}_i$ in terms of $\vec{f}_i$ from Equation 39, $\ddot{\vec{r}}_i(t) = \frac{1}{M}\vec{f}_i(t)$; substituting, we obtain:

$$\vec{r}_i(t+\delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\delta t) + \frac{1}{M}\vec{f}_i(t)(\delta t)^2 + O\Big((\delta t)^4\Big) \quad (43)$$

Equation 43 is known ad the Verlet algorithm. [34, Eq. (4)]

We can re-express the equation in terms of the velocities

$$\vec{v}_i(t) = \frac{\vec{r}_i(t+\delta t) - \vec{r}_i(t-\delta t)}{2\delta t} \quad (44)$$

$$-\vec{r}_i(t-\delta t) = \vec{v}_i(t)\cdot 2\delta t - \vec{r}_i(t+\delta t) \quad (45)$$

$$2\vec{r}_i(t+\delta t) = 2\vec{r}_i(t) + 2\vec{v}_i(t)\delta t + \frac{1}{M}\vec{f}_i(t)(\delta t)^2 + O\Big((\delta t)^4\Big) \quad (46)$$

$$\vec{r}_i(t+\delta t) = \vec{r}_i(t) + \vec{v}_i(t)\delta t + \frac{1}{2M}\vec{f}_i(t)(\delta t)^2 + O\Big((\delta t)^4\Big) \quad (47)$$

This expression gives access to the positions at time $t+\delta t$ with just the knowledge of positions, velocities, and forces at time t.

2.3.2 Thermostats

For a sufficiently large system, averages computed in the microcanonical ensemble, with fixed (N, V, E), are not much different from those computed in the canonical ensemble, with fixed (N, V, T). However, it may be desirable to be able to generate Molecular Dynamics (MD) trajectories that span the canonical ensemble, for example because one may want to control the temperature exactly. Moreover, some system may not be ergodic, with the harmonic system being an egregious example. This means that given an initial set of positions and momenta, $(\{\vec{r}^0\}, \{\vec{p}^0\})$, solving the Newton's equation of motion generates a trajectory that does not visit every neighbourhood of configurational space. In such a situation time averages are biased, and do not provide good approximations for ensemble averages.

One way to overcome this problem is to couple the simulated system with an external heat bath, provided that all degrees of freedom are interacting with the bath. Several techniques have been developed, but not all of them are capable of overcoming the ergodicity problem. One that does is the thermostat developed by Andersen, [35] which is based on the concept of stochastic collisions. We know that is a perfect gas at some temperature T the velocities are distributed according to the Maxwell distribution

$$f(v)\,\mathrm{d}v = \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} 4\pi v^2 \exp\left(-\frac{mv^2}{2k_B T}\right)\mathrm{d}v. \tag{48}$$

Therefore, one way to generate the canonical ensemble is to perform a MD simulation by repeatedly drawing velocities from a Maxwell distribution. This periodic velocity re-initialization procedure also redistributes energy between different modes, and so it is an effective way to overcome the ergodicity problem. It can be shown that the frequency of these velocity re-initializations does not affect the ability to sample the canonical ensemble; however, drawing the velocities too often will result in the system moving very slowly from one region of configuration space to another; drawing them too seldom results in slow transfer of energy between different modes, which would only overcome the ergodicity problem slowly. Finding the appropriate time interval between velocities randomizations is then a matter of finding the right compromise to maximize efficiency.

In **Andersen dynamics**, constant temperature is imposed by stochastic collisions with a heath bath. With a (small) probability the collisions act occasionally on velocity components of randomly selected particles. Upon a collision the new velocity is drawn from the Maxwell-Boltzmann distribution at the corresponding temperature. The system is then integrated numerically at constant energy according to the Newtonian laws of motion. The collision probability is defined as the average number of collisions per atom and timestep. The algorithm generates a canonical distribution. [36, §6.1.1] However, due to the random decorrelation of velocities, the dynamics are unphysical and cannot represent dynamical properties like e.g. diffusion or viscosity. Another disadvantage is that the collisions are stochastic in nature, so repeating the simulation will not give exactly the same trajectory. Typical values for the collision probability are in the order of $10^{-4} \div 10^{-1}$.

In **Nosé-Hoover dynamics**, an extra term is added to the Hamiltonian representing the coupling to the heat bath. From a pragmatic point of view one can regard Nosé-Hoover dynamics as adding a friction term to Newton's second law, but dynamically changing the friction coefficient to move the sys-

tem towards the desired temperature. Typically the "friction coefficient" will fluctuate around zero.

During simulations in the present work, the Langevin thermostat[1] was used for constant (N, V, T) MD and combined Nose-Hoover and Parrinello-Rahman[2] dynamics for the (N, P, T) ensemble. The Berendsen thermostat was considered, but later discarded[3] in favour of the thermostats above.

2.4 THERMODYNAMICS OF THE STABILITY OF CRYSTALS

The main quantity to consider to assess the stability of a crystal is its lattice energy, E_{latt}, which is the energy per molecule gained upon assuming the crystal form with respect to the gas state. It can be computed as:

$$E_{\text{latt}} = E_{\text{crys}} - E_{\text{gas}}, \tag{49}$$

with E_{crys} as the energy per molecule in the crystal state and E_{gas} as the energy of the isolated molecule.

Typically, the computation of E_{latt} is performed at zero temperature and considering only the electronic contribution, i.e., quantum nuclear effects are neglected. [37] The lattice energy is not directly assessable experimentally, but it can be indirectly obtained from experimental measures of the sublimation enthalpy, $\Delta_{\text{sub}}H(T)$, at a given temperature T, by including a (theoretically evaluated) energy contribution, $\Delta_{\text{T\&QN}}(T)$, accounting for thermal and quantum nuclear effects:

$$\Delta_{\text{sub}}H(T) = -E_{\text{latt}} + \Delta_{\text{T\&QN}}(T). \tag{50}$$

The evaluation of $\Delta_{\text{T\&QN}}(T)$ can be challenging, especially for large molecules where anharmonic contributions are important. [24] Since both $\Delta_{\text{sub}}H(T)$ and $\Delta_{\text{T\&QN}}(T)$ are affected by errors, accurate theoretical evaluations of E_{latt} are of help for comparison.

In order to derive $\Delta_{\text{T\&QN}}$, we need to start from the definition of the sublimation enthalpy, $\Delta_{\text{sub}}H(T)$, that is the difference between the enthalpy of the gas, $H^g(T)$, and of the crystal solid, $H^s(T)$, both at temperature T. By separating the electronic (el), translational (trans), rotational (rot) and vibrational (vib) contributions; noticing that in the crystal there are no trans-rotational contributions; considering as negligible the pressure times volume term, pV; we have that

[1] https://wiki.fysik.dtu.dk/ase/ase/md.html#module-ase.md.andersen

[2] https://wiki.fysik.dtu.dk/ase/ase/md.html#module-ase.md.npt

[3] See the "Flying ice cube" effect.

$$\Delta_{\text{sub}}H = E^g_{\text{el}} + E^g_{\text{trans}} + E^g_{\text{rot}} + E^g_{\text{vib}} + pV - (E^s_{\text{el}} + E^s_{\text{vib}}), \tag{51}$$

where the superscript stands either for gas (g) or solid (s), and the temperature dependance has been dropped for the seek of brevity. By assuming that the rigid rotor and ideal gas approximations are reliable (that is typically the case in the analyzed molecular systems), we have that $E^g_{\text{trans}} = \frac{3}{2}RT$, $E^g_{\text{rot}} = \frac{3}{2}RT$ if the molecule is non-linear, $E^g_{\text{rot}} = RT$ otherwise, and $pV = RT$. Thus, Equation 51 simplifies to

$$\begin{aligned} \Delta_{\text{sub}}H(T) &= \Delta E_{\text{el}} + \Delta E_{\text{vib}}(T) + 4RT \quad \text{for non-linear molecules,} \\ \Delta_{\text{sub}}H(T) &= \Delta E_{\text{el}} + \Delta_{\text{vib}}(T) + \frac{7}{2}RT \qquad \text{for linear molecules,} \end{aligned} \tag{52}$$

where the term $\Delta E_{\text{vib}}(T)$ contains both the thermal and the quantum nuclear contributions. Notice from that $\Delta E_{\text{el}} = E^g_{\text{el}} - E^s_{\text{el}}$ is precisely the opposite of the lattice energy E_{latt}; thus:

$$\begin{aligned} \Delta_{\text{T\&QN}}(T) &= \Delta E_{\text{vib}}(T) + 4RT \quad \text{for non-linear molecules,} \\ \Delta_{\text{T\&QN}}(T) &= \Delta E_{\text{vib}}(T) + \frac{7}{2}RT \qquad \text{for linear molecules.} \end{aligned} \tag{53}$$

Vibrations in the solid molecular crystals can usually be separated into intra-molecular and inter-molecular vibrations, $E^s_{\text{vib}} = E^{s,\text{intra}}_{\text{vib}} + E^{s,\text{inter}}_{\text{vib}}$, and the stiffest intra-molecular modes are decoupled from the intermolecular modes. Intra-molecular vibrations have similar modes and frequencies than the gas-phase molecule; thus we can conveniently write:

$$\Delta E_{\text{vib}} = \Delta E^{\text{relax}}_{\text{vib}} - E^{s,\text{inter}}_{\text{vib}} \tag{54}$$

where $\Delta E^{\text{relax}}_{\text{vib}} \coloneqq E^g_{\text{vib}} - E^{s,\text{intra}}_{\text{vib}}$ is the change in (intra-molecular) vibrational energy given when molecules are packed in the crystal form.

At this point, a first approach can be to do a drastic approximation (which is anyway often found in the literature), that is to assume that intra-molecular frequencies in the solid are exactly the same as in the gas phase (i.e. $\Delta E^{\text{relax}}_{\text{vib}} \approx 0$), then to take the high temperature limit for the inter-molecular vibrations (i.e. $E^{s,\text{intra}}_{\text{vib}} \approx 6RT$) and to neglect any zero-point motion, yielding $\Delta E_{\text{vib}}(T) \approx -6RT$ (Dulong-Petit law). In non-linear molecules, that would imply that $\Delta_{\text{T\&QN}} \approx -2RT$, that is 4.96 kJ/mol at room temperature, $T = 298.15K$, and zero at $T = 0K$. This is a poor approximation. [38] This approximation is particularly bad for water ice. [39, SI §12.B], [40]

A more reliable approach is to calculate the vibrational energies for the solid and gas phase in the harmonic limit, considering for each frequency ω a contribution

$$\varepsilon(\omega, T) = \frac{\hbar\omega}{2} + \frac{\hbar\omega}{\exp\left(\frac{\hbar\omega}{k_B T}\right) - 1}, \tag{55}$$

where the first term in the right hand side accounts for the Zero Point Energy (ZPE) contribution and the second for the thermal one.

This yields

$$E^g_{\text{vib}}(T) = \sum_i \varepsilon(\omega_i, T), \quad E^s_{\text{vib}}(T) = \int \varepsilon(\omega, T) g(\omega)\, \mathrm{d}\omega, \tag{56}$$

where ω_i are the frequencies of the isolated molecule, which are $3M - 6$ (M is the number of atoms in the molecule) for a non-linear molecule, and $3M - 5$ for a linear molecule; $g(\omega)$ is the phonon density of states in the solid. [38], [24]

Notice that whenever we employ Equation 56 to evaluate $\Delta_{\text{T\&QN}}$ in Equation 53, we are subject to errors not only coming from the harmonic approximation, but also from the limitations of the computational approaches (typically DFT) used for the evaluations of the frequencies and the phonon spectrum. Different choices of the exchange-correlation functional in DFT can lead to differences in terms of $\Delta_{\text{T\&QN}}$ quite larger than 1 kJ/mol. In particular, inaccuracies on the evaluation of high frequency modes mostly affects the ZPE contribution, while low frequencies modes affect mostly the thermal contribution.

2.4.1 Dispersion interactions

Dispersion interactions are essential in a collection of active research fields in solid-state physics and chemistry, including molecular crystal packing, crystal structure prediction, surface adsorption and reactivity, and supramolecular chemistry. The representation of dispersion interactions in DFT is not possible within local or semilocal functionals because dispersion arises from non-local correlation effects involving distant fragments in the crystal. [38]

The Exchange-hole Dipole Moment (XDM) model describes the dispersion energy of two neutral fragments as the electrostatic interaction of the dipoles formed by electrons and their associated exchange holes. The dispersion energy is added to the DFT energy

$$E = E_{\text{DFT}} + E_{\text{disp}}, \tag{57}$$

where E_{disp} contains the usual R^{-6} leading term as well as two additional higher order atomic-pairwise terms

$$E_{\text{disp}} = -\frac{1}{2}\sum_{ij}\sum_{n=6,8,10}\frac{C_{n,ij}}{R^n_{\text{vdw},ij} + R^n_{ij}}. \tag{58}$$

The fundamental objects in this equation are the inter-atomic interaction coefficients $C_{n,ij}$ that in the XDM model are calculated exclusively from first-principles quantities using second-order perturbation theory.

All the objects above are parameter-free, except for the damping expression in . The interatomic van der Waals radii ($R_{\text{vdw},ij}$) control the distance at which the pairwise dispersion interactions are switched off.

Because the dispersion coefficients are calculated rather than fitted, Equation 57 works under the assumption that the DFT functional presents a completely dispersionless behavior. This requirement is not met by most Generalized Gradient Approximation (GGA) functionals, which are sometimes too repulsive and sometimes spuriously binding, depending on the reduced-density-gradient tail behavior of the exchange enhancement factors.

2.4.1.1 DFT-D3

The total DFT-D3 energy is given by [41]

$$E_{\text{DFT-D3}} = E_{\text{KS-DFT}} - E_{\text{disp}}, \tag{59}$$

where $E_{\text{KS-DFT}}$ is the usual self-consistent KS energy as obtained from the chosen DF and E_{disp} is the dispersion correction as a sum of two- and three-body energies,

$$E_{\text{disp}} = E^{(2)} + E^{(3)}, \tag{60}$$

with the most important two-body term given by

$$E^{(2)} = \sum_{AB}\sum_{n=6,8,10,\ldots} s_n \frac{C_n^{AB}}{r_{AB}^n} f_{d,n}(r_{AB}). \tag{61}$$

Here, the first sum is over all atom pairs in the system, C_n^{AB} denotes the averaged (isotropic) nth-order dispersion coefficient (orders $n = 6, 8, 10, ...$) for atom pair AB, and r_{AB} is their internuclear distance. $f_{d,n}$ are damping functions explicitly chosen by original authors to make the model numerically stable.

2.5 MACHINE LEARNING

Machine Learning can be described as *the application and science of algorithms that make sense of data.* [4, §1] There are three types of machine learning: su-

pervised learning, unsupervised learning, and reinforcement learning. It is our interest to study the **supervised learning** type. The main goal in supervised learning is to learn a model from labeled training data that allows us to make predictions about unseen or future data. Here, the term "supervised" refers to a set of training examples (data inputs) where the desired output signals (labels) are already known. Supervised learning is then the process of modeling the relationship between the data inputs and the labels. Thus, we can also think of supervised learning as "label learning". A supervised learning task with discrete class labels is also called a **classification task**. Another subcategory of supervised learning is **regression**, where the outcome signal is a continuous value. In the detailed description of MACE in Section 5.3, we will see that the data inputs are atom positions, atomic number; while the label will be the energy, a continuous value learned through regression. In regression analysis, we are given a number of predictor (**explanatory**) variables and a continuous response variable (**outcome**), and we try to find a relationship between those variables that allows us to predict an outcome. In the field of machine learning, the predictor variables are commonly called "features", and the response variables are usually referred to as "target variables".

Artificial neurons represent the building blocks of the multilayer artificial Neural Networks (NNs). The basic concept behind artificial NNs was built upon hypotheses and models of how the human brain works to solve complex problem tasks. NNs are more popular today than ever thanks to the many breakthroughs that have been made in the previous decade, which resulted in what we now call deep learning algorithms and architectures—NNs that are composed of many layers.

2.5.1 Single layer neural network

Before we dig deeper into a particular multilayer NN architecture, let's briefly reiterate some of the concepts of single-layer NNs, namely, the ADAptive LInear Neuron (Adaline) algorithm.

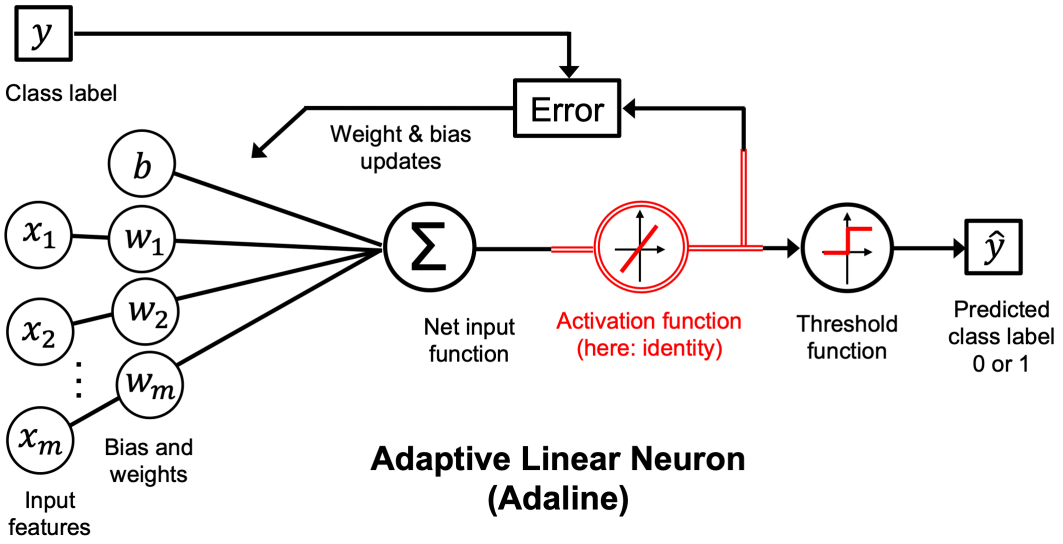


Figure 4: The Adaline algorithm. Taken from [4, Fig. 11.1].

The Adaline algorithm performs binary classification, and uses the gradient descent optimization algorithm to learn the weight coefficients of the model. In every epoch (pass over the training dataset), we update the weight vector $\vec{w}$ and bias unit b using the following update rule:

$$\vec{w} := \vec{w} + \Delta\vec{w}, \quad b := b + \Delta b, \tag{62}$$

where $\Delta w_j = -\eta \frac{\partial L}{\partial w_j}$ for each weight w_j in the weight vector $\vec{w}$ and $\Delta b = -\eta \frac{\partial L}{\partial b}$ for the bias unit. In other words, we computed the gradient based on the whole training dataset and updated the weights of the model by taking a step in the opposite direction of the loss gradient $\nabla L(\vec{w})$. In order to find the optimal weights of the model, we optimize an objective function that we define as the mean of squared errors (MSE) loss function $L(\vec{w})$. Furthermore, we multiply the gradient by a factor, the learning rate η, which we have to choose carefully to balance the speed of learning against the risk of overshooting the global minimum of the loss function. In gradient descent optimization, we update all weights simultaneously after each epoch, and we define the partial derivative for each weight w_j in the weight vector, $\vec{w}$, as follows:

$$\frac{\partial L}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{n} \sum_i \left(y^{(i)} - a^{(i)}\right)^2 = -\frac{2}{n} \sum_i \left(y^{(i)} - a^{(i)}\right) x_j^{(i)}. \tag{63}$$

Here, $y^{(i)}$ is the target class label of a particular sample $x^{(i)}$, and $a^{(i)}$ is the activation of the neuron, which is a linear function in the special case of Adaline. Furthermore, one can define the activation function $\sigma(\cdot)$ as follows:

$$\sigma(\cdot) = z = a. \tag{64}$$

Here, the net input, z, is a linear combination of the weights that are connecting the input layer to the output layer:

$$z = \sum_j w_j x_j + b = \vec{w}^\top \vec{x} + b. \tag{65}$$

While the activation $\sigma(\cdot)$ is used to compute the gradient update, a threshold function is implemented to squash the continuous-valued output into binary class labels for prediction:

$$\hat{y} = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{66}$$

A frequent technique used to accelerate the model training is the so-called **stochastic gradient descent (SGD)** optimization. SGD approximates the loss from a single training example (online learning) or a small subset of training examples (mini-batch learning). Apart from faster learning—due to the more frequent weight updates compared to gradient descent—its noisy nature is also regarded as beneficial when training multilayer NNs with nonlinear activation functions, which do not have a convex loss function. Here, the added noise can help to escape local loss minima.

2.5.2 The multilayer neural network architecture

Here we will explain how to connect multiple single neurons to a multilayer feedforward NN; this special type of *fully connected* network is also called Multi-Layer Perceptron (MLP).

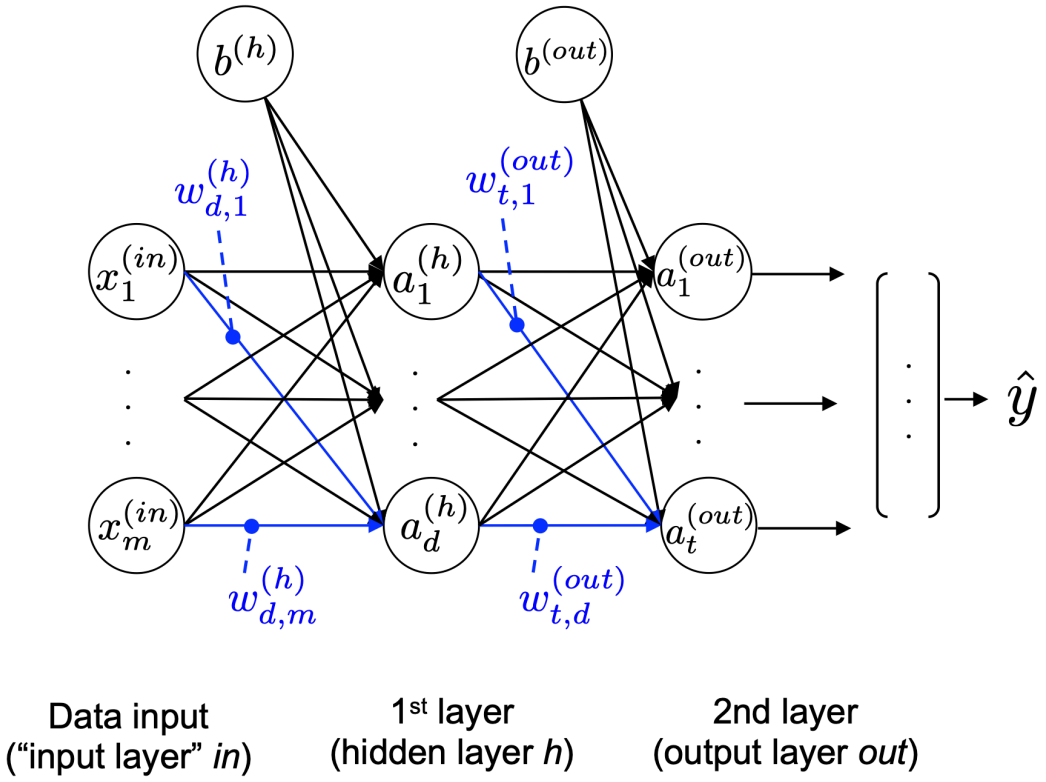


Figure 5: A two-layer MLP. Figure from [4].

Next to the data input, the MLP depicted in Figure 5 has one hidden layer and one output layer. The units in the hidden layer are fully connected to the input features, and the output layer is fully connected to the hidden layer. If such a network has more than one hidden layer, we alco call it a **deep NN**. The number of layers and units in a NN can be tought of as additional hyper-parameters that we want to optimize for a given problem.

We denote the ith activation unit in the lth layer as $a_i^{(l)}$. We use the "in" superscript for the input features, the "h" superscript for the hidden layer, and the "out" superscript for the output layer. The $\vec{b}$'s denote bias units; those are vectors with the number of elements being equal to the number of nodes in the layer they correspond to.

Each node in layer l is connected to all nodes in layer $l+1$ via a weight coefficient. The connection between the kth unit in layer l to the jth unit in layer $l+1$ will be written as $w_{j,k}^{(l)}$. We denote the weight matrix that connects the input to the hidden layer as $W^{(h)}$, and we write the matrix that connects the hidden layer to the output layer as $W^{(\text{out})}$.

The usage of multiple units in the output layer allow for native multi-class classification, via a generalization of the one-versus-all (OvA) technique. This is similar to the one-hot representation of categorical variables.

To calculate the output of a MLP model, we employ the process of **forward propagation**. The MLP learning procedure can be summarized in three steps:

1. Starting at the input layer, forward propagate the patterns of the training data through the network to generate an output.
2. Based on the network's output, calculate the loss that we want to minimize using a loss functions of choice.
3. Backpropagate the loss, find its derivative with respect to each weight and bias unit in the network, and update the model.

Finally, after we repeat these three steps for multiple epochs and learn the weight and bias parameters of the MLP, we use forward propagation to calculate the network output.

Since each unit in the hidden layer is connected to all units in the input layer, we first calculate the activation unit of the hidden layer $a_1^{(h)}$ as follows:

$$\begin{aligned} z_1^{(h)} &= x_1^{(\text{in})} w_{1,1}^{(h)} + x_2^{(\text{in})} w_{1,2}^{(h)} + \ldots + x_m^{(\text{in})} w_{1,m}^{(h)}, \\ a_1^{(h)} &= \sigma\left(z_1^{(h)}\right). \end{aligned} \tag{67}$$

Here, $z_1^{(h)}$ is the net input and $\sigma(\cdot)$ is the activation function, which has to be differentiable to learn the weights that connect the neurons using a gradient-based approach. To be able to solve complex problems, the MLP model needs nonlinear activation functions, for example, the sigmoid (logistic) activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{68}$$

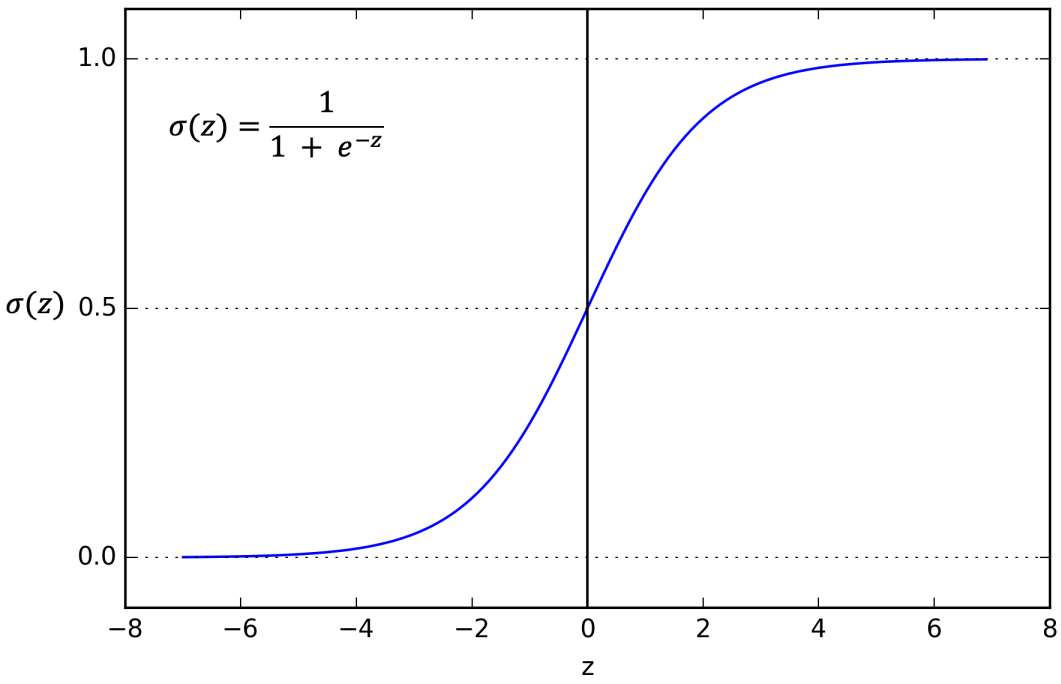


Figure 6: The sigmoid activation function. Figure from [4].

The MLP is a typical example of a feedforward artificial NN. The term **feed-forward** refers to the fact that each layer serves as the input to the next layer without loops, in contrast to recurrent NNs.

The activation is usually written in a more compact, vectorized form, using the concepts of linear algebra:

$$\begin{aligned} \vec{z}^{(h)} &= \vec{x}^{(\text{in})} W^{(h)\mathsf{T}} + \vec{b}^{(h)}, \\ \vec{a}^{(h)} &= \sigma\big(\vec{z}^{(h)}\big). \end{aligned} \tag{69}$$

Here, $\vec{z}^{(h)}$ is a $1 \times m$ dimensional feature vector; $W^{(h)}$ is a $d \times m$ dimensional weight matrix, where d is the number of units in the hidden layer; the bias vector $\vec{b}^{(h)}$ consists of d bias units (one bias unit per node).

Generalizing the computation to all n samples in the training dataset:

$$Z^{(h)} = X^{(\text{in})} W^{(h)\mathsf{T}} + \vec{b}^{(h)} \tag{70}$$

Here, $X^{(\text{in})}$ is a $n \times m$ matrix, and the matrix multiplication will result in a $n \times d$ dimensional net input matrix, $Z^{(h)}$. Finally, we apply the activation function $\sigma(\cdot)$ to each value in the net input matrix to get the $n \times d$ dimensional activation matrix in the next layer:

$$A^{(h)} = \sigma\big(Z^{(h)}\big) \tag{71}$$

Similarly for the output layer we get:

$$Z^{(\text{out})} = A^{(h)} W^{(\text{out})\mathsf{T}} + \vec{b}^{(\text{out})}, \quad A^{(\text{out})} = \sigma\big(Z^{(\text{out})}\big). \tag{72}$$

2.5.3 Training an artificial neural network

We use an MSE loss (as in Adaline) to train the multilayer NN as it makes the derivation of the gradients a bit easier to follow. If we predict the class label of an input data with class label 2, using this MLP, the activation of the third layer and the target may look like this:

$$a^{(\text{out})} = \begin{pmatrix} 0.1 \\ 0.9 \\ \vdots \\ 0.3 \end{pmatrix}, \quad y = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}. \tag{73}$$

Thus, our MSE loss either has to sum or average over the t activation units in our network in addition to averaging over the n examples in the dataset or mini-batch:

$$L\big(W, \vec{b}\big) = \frac{1}{n} \sum_{1}^{n} \frac{1}{t} \sum_{j=1}^{t} \left(y_j^{(i)} - a_j^{(\text{out})(i)}\right)^2 \tag{74}$$

The goal is to minimize the loss function $L(W)$. We need to calculate the partial derivative of the parameters W with respect to each weight for every layer in the network:

$$\frac{\partial L}{\partial w_{j,l}^{(i)}} \tag{75}$$

Note that W consists of multiple matrices. In an MLP with one hidden layer, we have the weight matrix, $W^{(h)}$, which connects the input to the hidden layer, and $W^{(\text{out})}$, which connects the hidden layer to the output layer.

Backpropagation is a very efficient and one of the most widely used algorithms for training artificial NNs. In essence, we can think of backpropagation as a very computationally efficient approach to compute the partial derivatives of a complex, non-convex loss function in multilayer NNs. The goal is to use those derivatives to learn the weight coefficients for parameterizing such a multilayer artificial NN. The error surface of an NN loss function is not convex or smooth with respect to the parameters. There are many bumps in this high-dimensional loss surface (local minima) that we have to overcome in order to find the global minimum of the loss function. Given the function $F(x) = f(g(h(u(v(x)))))$, we can use the chain rule to compute the derivative:

$$\frac{\mathrm{d}F}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x} f(g(h(u(v(x))))) = \frac{\mathrm{d}f}{\mathrm{d}g}\frac{\mathrm{d}g}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}v}\frac{\mathrm{d}v}{\mathrm{d}x}. \tag{76}$$

In the context of computer algebra, a set of techniques, known as **automatic differentiation**, has been developed to solve such problems very efficiently.

Automatic differentiation comes with two modes, the forward and reverse modes; backpropagation is simply a special case of reverse-mode automatic differentiation. The key point is that applying the chain rule in forward mode could be quite expensive since we would have to multiply large matrices for each layer (Jacobians) that we would eventually multiply by a vector to obtain the output.

The trick of reverse mode is that we traverse the chain rule from right to left. We multiply a matrix by a vector, which yields another vector that is multiplied by the next matrix, and so on. Matrix-vector multiplication is computationally much cheaper than matrix-matrix multiplication, which is why backpropagation is one of the most popular algorithms used in NN training.

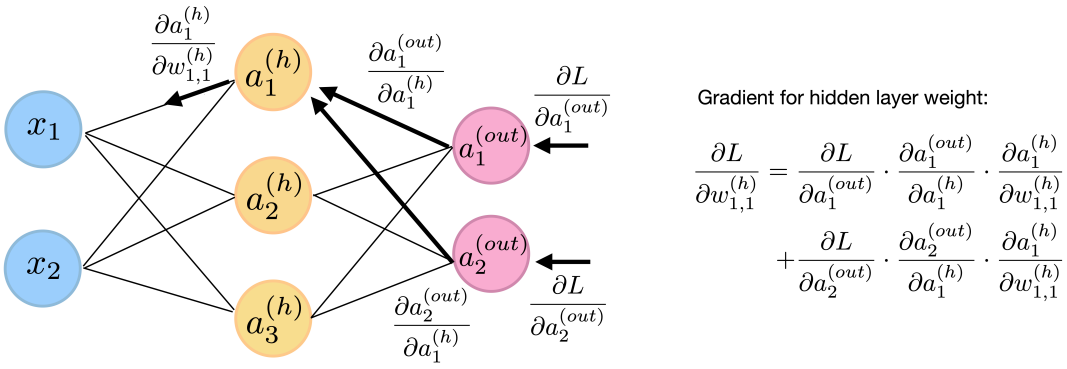


Figure 7: Computing the partial derivatives of the loss with respect to the first hiddel layer weight. Averaging over the mini-batch is omitted. Figure from [4].

2.6 GRAPH NEURAL NETWORKS

Neural networks have been adapted to leverage the structure and properties of graphs.

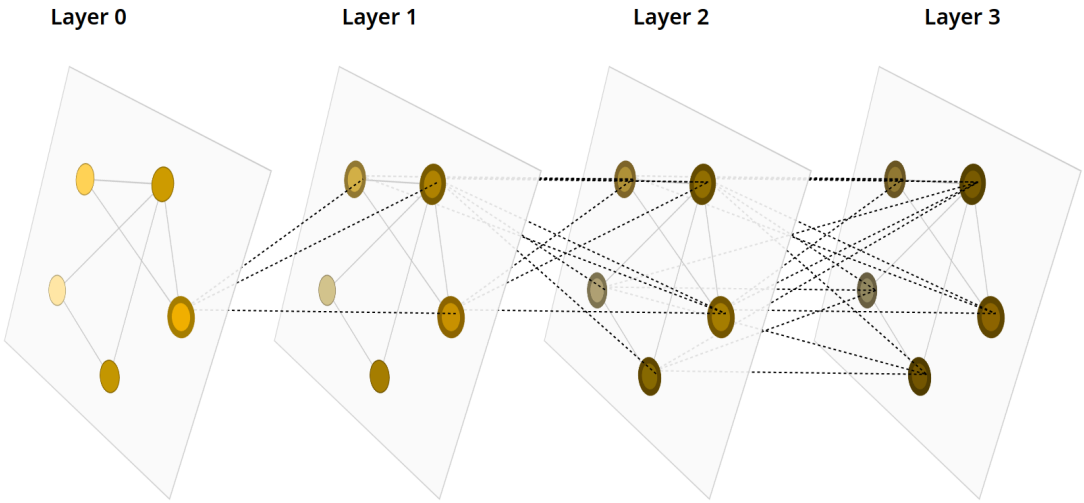
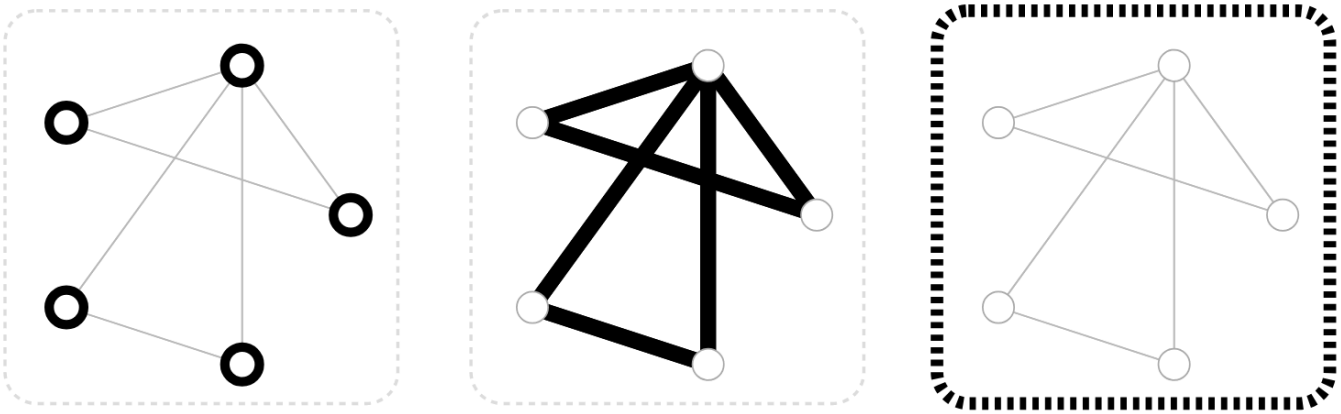


Figure 8: Taken from [5]. Diagram representing of how a node accumulates information from nodes around it through the layers of the network.

A set of objects, and the connections between them, are naturally expressed as a graph. Graph Neural Networks (GNNs) see practical applications in physics simulations [42] and are the foundation of the main calculator employed in this work, MACE, as detailed in Section 5.3.

A graph represents the relations (*edges*) between a collection of entities (*nodes*).



V – Vertex (or node) attributes e.g., node identity, number of neighbors.

E – Edge (or link) attributes and directions e.g., edge identity, edge weight.

U – Global (or master node) attributes e.g., number of nodes, longest path.

Figure 9: Three types of attributes we might find in a graph. Figures from [5].

To further describe each node, edge or the entire graph, we can store information in each of these pieces of the graph. We can additionally specialize graphs by associating directionality to edges (*directed*, *undirected*).

Graphs are very flexible data structures, and for this reason they were used by MACE to embed atomistic properties of physical systems. It's a very convenient and common abstraction to describe this 3D object as a graph, e.g. where nodes are atoms and edges are covalent bonds. A way of visualizing the connectivity of a graph is through its adjacency matrix. One labels the nodes, in this case each of 14 non-H atoms in a caffeine molecule, and fill a matrix of $n_{\text{nodes}} \times n_{\text{nodes}}$ with an entry if two nodes share an edge.

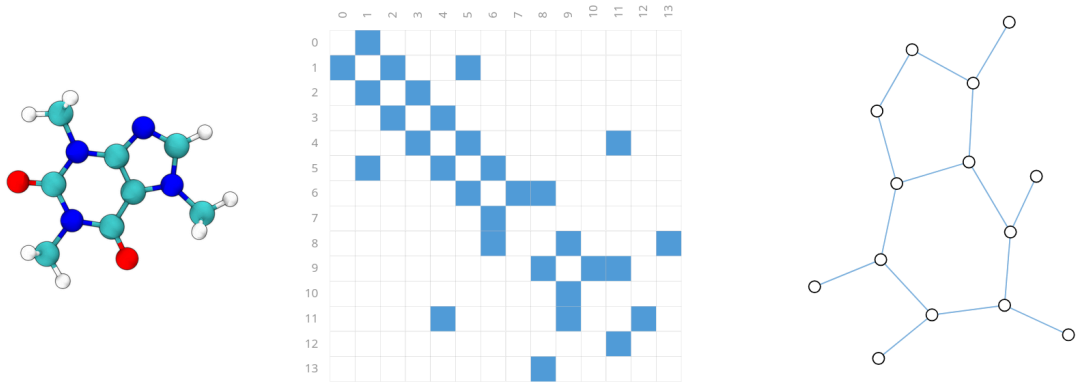


Figure 10: Figure from [5]. (Left) 3D representation of the Caffeine molecule. (Center) Adjacency matrix of the bonds in the molecule. (Right) Graph representation of the molecule.

Definition: A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances).

In the following, we will describe the Message Passing Neural Network (MPNN) framework proposed by [3] using the Graph Nets architecture schematics introduced by [43]. GNNs adopt a "graph-in, graph-out" architecture meaning that these model types accept a graph as input, with information loaded into its nodes, edges and global-context, and progressively transform these embeddings, without changing the connectivity of the input graph.

The GNN uses a differentiable model of choice (e.g. a multilayer perceptron (MLP)) on each component of a graph; this is a GNN layer. For each node vector, one applies the model and gets back a learned node-vector. One does the same for each edge, learning a per-edge embedding, and also for the global-context vector, learning a single embedding for the entire graph.

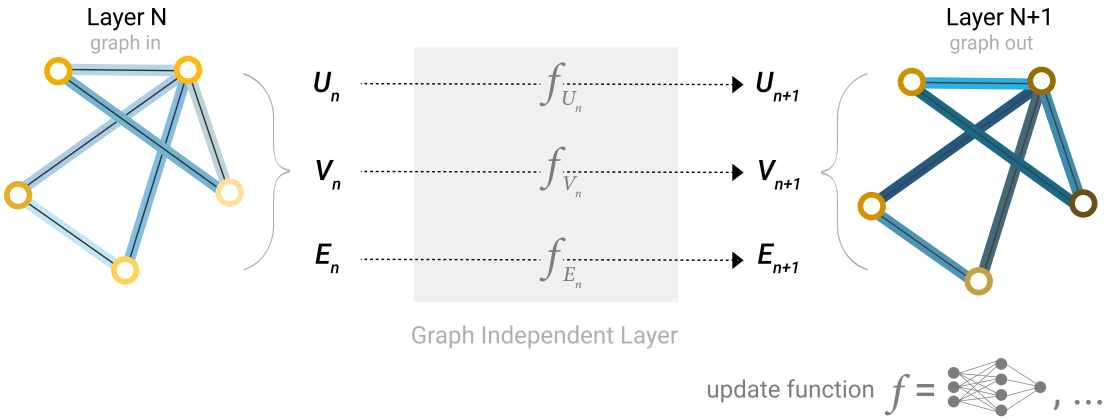


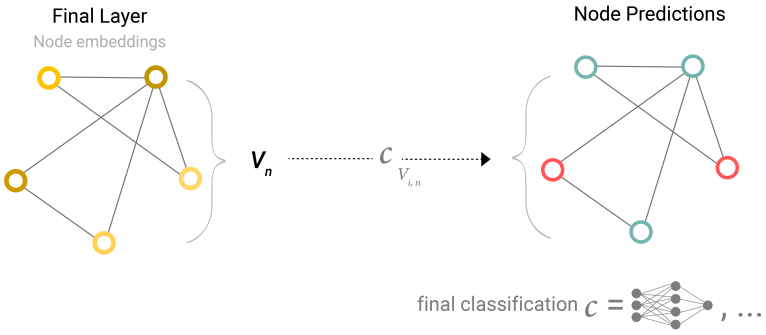
Figure 11: A single layer of a simple GNN. A graph is the input, and each component (V, E, U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n-th layer of a GNN model.

As is common with neural network modules or layers, one can stack these GNN layers together.

Because a GNN does not update the connectivity of the input graph, one can describe the output graph of a GNN with the same adjacency list and the same number of feature vectors as the input graph. But, the output graph has updated embeddings, since the GNN has updated each of the node, edge and global-context representations.

2.6.1 GNN Predictions by Pooling Information

How does a GNN make predictions in any of its tasks described above? Prediction tasks can belong to binary classification, multi-class classification or regression cases. In the following example, the binary classification will be considerer for brevity, but this framework extends to the other cases. If the task is to make predictions on nodes, and the graphs already contains node information, the approach is straightforward—for each node embedding, apply a linear classifier.



However, it is not always so simple. For instance, one might have information in the graph stored in edges, and no information in nodes,but still need to make predictions on nodes. One needs a way to collect information from edges and give them to nodes for prediction. One can do this by *pooling*. Pooling proceeds in two steps:

1. For each item to be pooled, *gather* each of their embeddings and concatenate them into a matrix.
2. The gathered embeddings are then *aggregated*, usually via a sum operation.

[5] represents the *pooling* operation by the letter ρ, and denotes that we are gathering information from edges to nodes as $p_{E_n \to V_n}$.

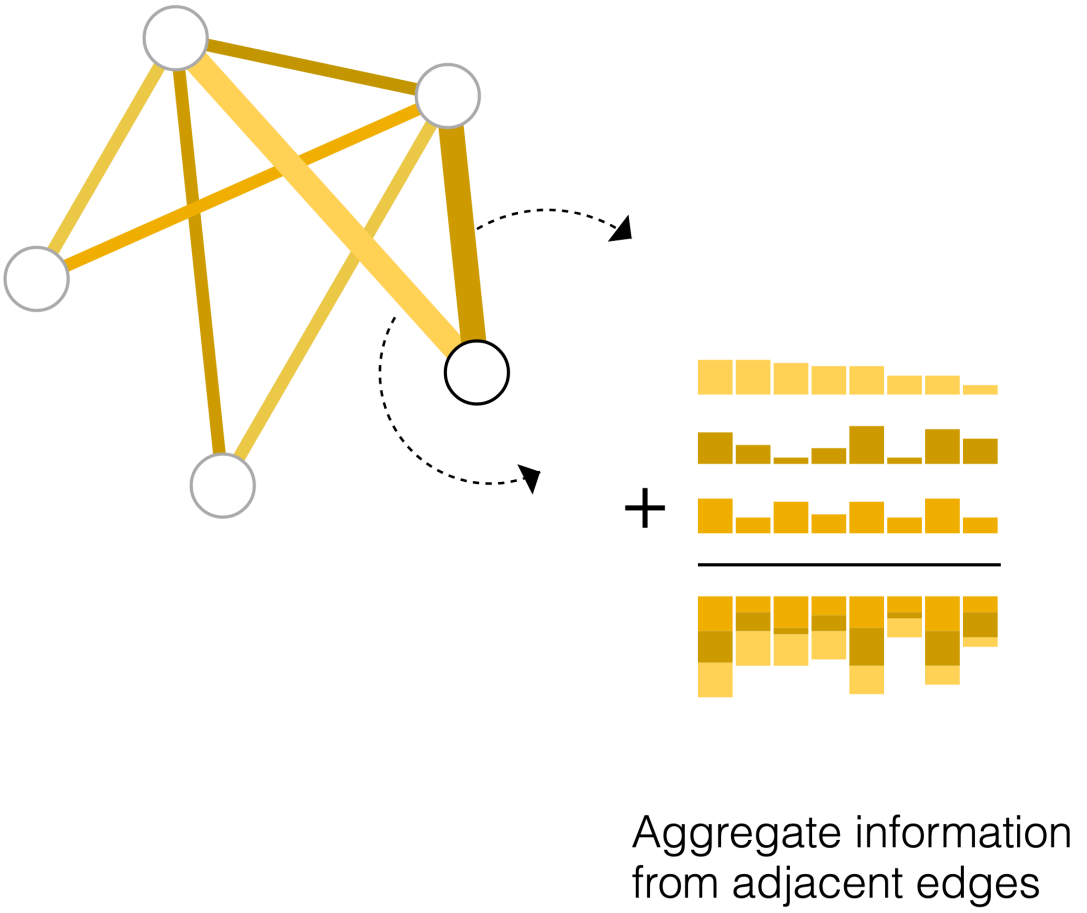


Figure 13: The edges connected to the black node are gathered and aggregated to produce an embedding for that target node. Figure from [5].

So if we only have edge-level features, and are trying to predict node information, we can use pooling to route (or pass) information to where it needs to go, the model looks like this:

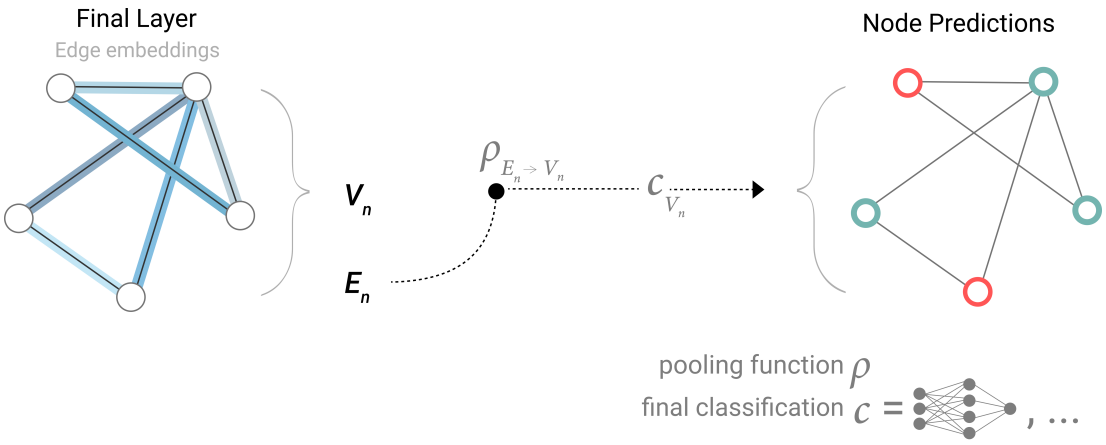


Figure 14: Figure from [5].

The same reasoning goes for the prediction of edge-level information and global properties, gathering available node and/or edge information together and aggregating them to get the desired predictions. This technique is similar to *Global Average Pooling* layers in Convolutional Neural Networks (CNNs).

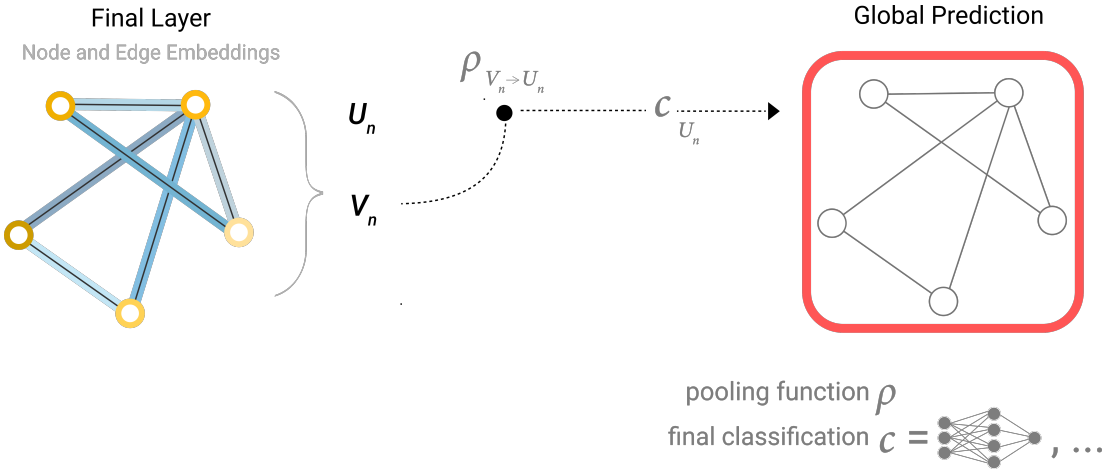


Figure 15: This is a common scenario for predictiong molecular properties. Figure from [5].

The classification model $\mathcal{C}$ can easily be replaced with any differentiable model, or adapted to multi-class classification using a generalized linear model.

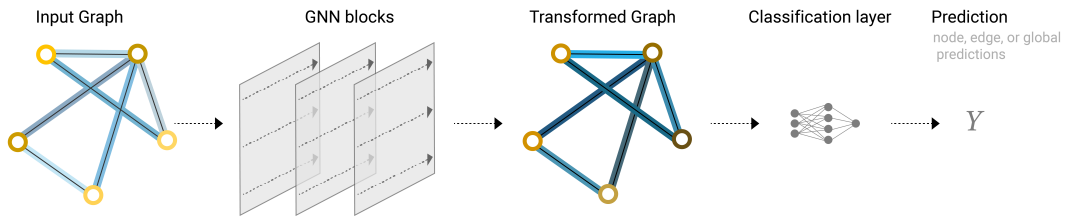


Figure 16: An end-to-end prediction task with a GNN model. Figure from [5].

2.6.2 Passing messages between parts of the graph

One could make more sophisticated predictions by using pooling within the GNN layer, in order to make the learned embeddings aware of graph connectivity. We can do this using *message passing* [3], where neighbouring nodes or edges exchange information and influence each other's update embeddings.

Message passing works in three steps:

1. For each node in the graph, *gather* all the neighbouring node embeddings (or messages).
2. Aggregate all messages via an aggregate function (like sum).
3. All pooled messages are passed through an *update function*, usually a learned neural network.

Just as pooling can be applied to either nodes or edges, message passing can occur between either nodes or edges.

These steps are key for leveraging the connectivity of graphs. One can build more elaborate variants of message passing in GNN layers that yield GNN models of increasing expressiveness and power.

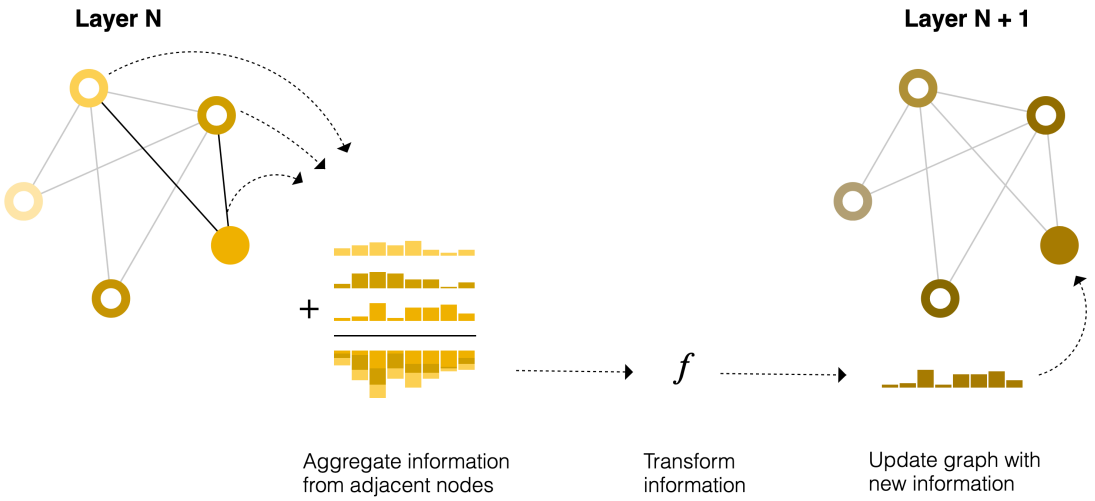


Figure 17: Pooling, update and storage of the adjacent embedding for the highlighted node. Figure from [5].

This sequence of operations, when applied once, is the simplest type of message-passing GNN layer. This is reminiscent of standard convolution: in essence, message passing and convolution are operations to aggregate and process the information of an element's neighbours in order to update the element's value. In graphs, the element is a node, and in images, the element is a pixel. However, the number of neighbouring nodes in a graph can be variable, unlike in an image where each pixel has a set number of neighbouring elements.

By stacking messge passing GNN layers together, a node can eventually incorporate information from across the entire graph: after three layers, a node has information about the nodes three steps away from it.

The updated architecture diagram to include this new source of information for nodes is the following:

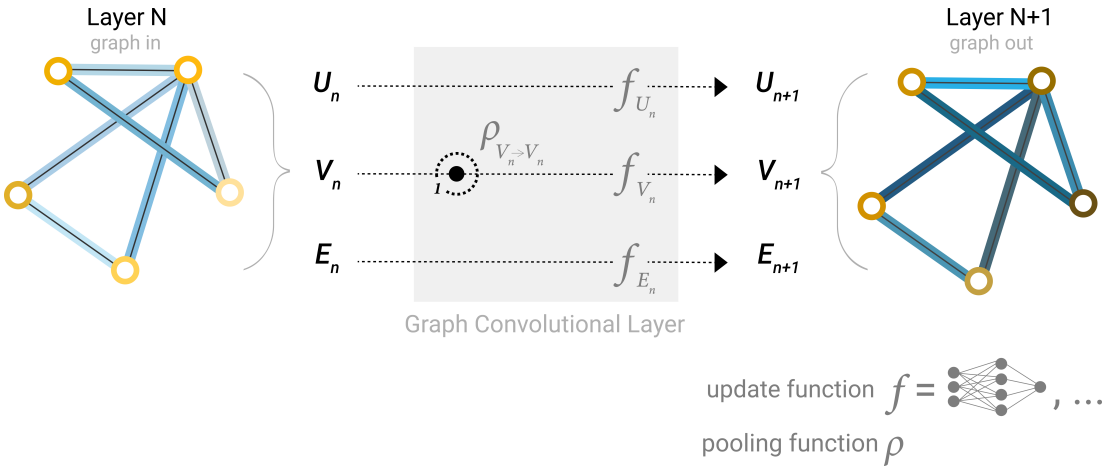


Figure 18: Schematic for a GCN architecture, which updates node representations of a graph by pooling neighbouring nodes at a distance of one degree. Figure from [5].

The notions exposed above are a sufficient introduction to understand the basic functioning of the MACE calculator, described in detail in Section 5.3, that was used for the work in this thesis, the results of which are available in the next chapters, Section 3 and Section 4.

3 | RESULTS I: MODEL ASSESSMENT

In this chapter we test the MACE calculators on small, known systems, about which literature is abundant and a direct comparison of results is possible. We analyze the geometrical and vibrational features of the water molecule and the water dimer after optimization with each calculator. The binding energy of the dimer is calculated and compared with reference DFT methods. This setting already allows to quantify the benefits of a fine-tuned MLP like MACE-ICE13-1 compared with MACE foundation models.

3.1 THE WATER MOLECULE

The first task is the optimization of the geometry and calculation of vibrational properties of the water molecule. We studied the relaxation of the geometry of a single water molecule, also referred to as the monomer, and compared the results with physical values of the gas phase molecule. The optimization is tackled with two concurring methods: static local **minimization of the potential energy** and **analysis of vibrational properties** to assess the dynamical stability.

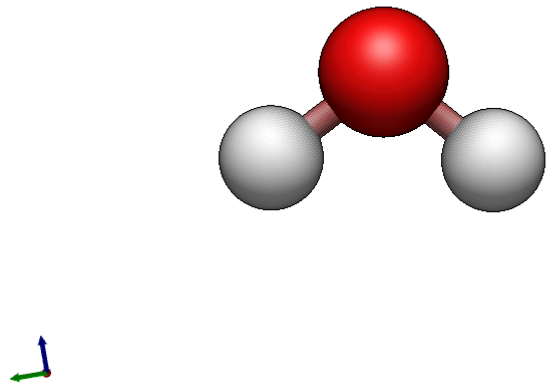


Figure 19: Render of the geometry of the water molecule, optimized using MACE-ICE13-1.

3.1.1 Geometry optimization

To rapidly put to the test the calculators, geometrical values of the relaxed configuration have been computed and compared with references in literature.

The first value concerns the charachteristic HOH **bend angle** of the molecule; the accurate description of this physical value is a required test to ensure the validity of the models. As can be seen in Table 1, the MACE-MP-0 large model most accurately describes the bend angle of the molecule, while the small model is the most distant from reference [6], [44].

The second value is the OH **bond length**. Table 2 shows that MACE-ICE13-1 gives the most accurate description according to reference [6], [44], and MACE-MP-0 small again is less accurate than the other models.

Model	HOH Angle (°)	Discrepancy (°)
small	105.652	1.129
medium	104.106	-0.417
large	104.494	-0.029
MACE-ICE13-1	103.970	-0.553
Reference	104.523	0.000

Table 1: The calculated HOH bend angle for the gas phase water monomer, and difference with respect to reference. small, medium and large refer to MACE-MP-0.

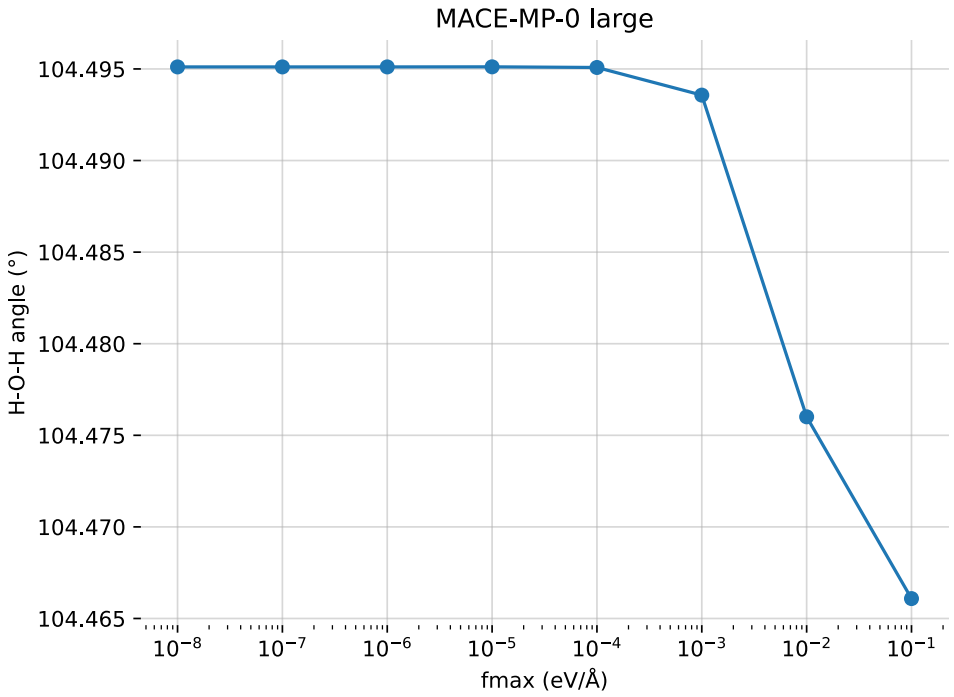


Figure 20: Convergence of the HOH angle with respect to the `fmax` parameter.

Model	Bond length (Å)	Discrepancy (Å)
small	0.97430	0.01712
medium	0.97296	0.01578
large	0.97267	0.01549
MACE-ICE13-1	0.96974	0.01256
Reference	0.95718	0.00000

Table 2: Calculated OH bond lengths for the gas phase water monomer, and difference with respect to reference.

The following sections detail the procedure to assess the convergence and stability of the obtained geometries, as well as explore the energetics of the molecule.

3.1.2 The optimization algorithm

The procedure to optimize the geometry of the molecule requires the correct setting of a few parameters, namely:

- the calculator and the data type
- the starting geometry and cell properties
- the optimizer algorithm
- the force threshold to stop the optimization

- the maximum number of steps of the optimization

The calculators chosen for the optimization are MACE-MP-0, in the small, medium and large variants, and MACE-ICE13-1. As advised by the calculators, the float64 data type was chosen for optimization, where float32 is advised for molecular dynamics runs for improved speed.

The initial geometry was set up so that the HOH atoms formed a right angle, and the OH distance was 1Å. By dealing with a force field, it was possible to set up a cell in vacuum without periodic boundary conditions.

Atomic Simulation Environment (ASE) offers a variety of different optimisers, from which we have chosen BFGS, a local optimisation algorithm of the quasi-Newton category, where the forces of consecutive steps are used to dynamically update a Hessian describing the curvature of the potential energy landscape. The optimiser accepts two important input parameters. The first if *fmax*, the force threshold, defined in eV Å^{-1}. The convergence criterion is that the force on all individual atoms should be less than fmax:[4]

$$\max_a |\vec{F}_a| < f_{\max} \tag{77}$$

For the present purposes, `fmax=1e-8` yielded satisfactory results for the different calculator models tested.

The second parameter is the number of steps after which to stop the optimization procedure. If the steps employed to optimize the geometry, given the desired fmax, are larger than the steps parameter, the procedure is halted and the geometry is considered as not converged. A value of `steps=1000` is largely above the actual number of steps required for convergence of the geometry.

```
from ase.optimize import BFGS
opt = BFGS(
  atoms,
  logfile="optimization.log",
  trajectory="optimization.traj"
)
opt.run(fmax=1e-8, steps=1000)
```

Listing 1: Initialisation and run of the optimizer.

3.1.3 Molecular vibrations and assessment of the dynamical stability

The nuclei of molecules, far from occupying fixed positions with respect to each other, are in continual state of vibration, even at 0°K. An important feature of these vibrations is that they can be described by a limited number of basic

[4] https://wiki.fysik.dtu.dk/ase/ase/optimize.html

vibrations known as the normal modes. A normal mode is a vibration in which all the nuclei oscillate with the same frequency and the same phase. The water molecule has three normal modes and every possible vibration of the molecule can be described as a superposition of these three modes.

The normal modes of vibration of water are shown in Figure 21. Because the motion of the nuclei in the ν_1 and ν_3 vibrations is nearly along the direction of the $O-H$, these modes are often referred to as $O-H$ stretching vibrations. Similarly, because the H nuclei in ν_2 move in directions almost perpendicular to the bonds, ν_2 is referred to as the $H-O-H$ bending vibration. In fact, ν_1 involves a small amount of $H-O-H$ bending, and ν_2 involves a small amount of $O-H$ stretching. The mode ν_3 is called the asymmetric stretching vibration to distinguish it from the symmetric stretching vibration ν_1. [6, §1.1 (d)]

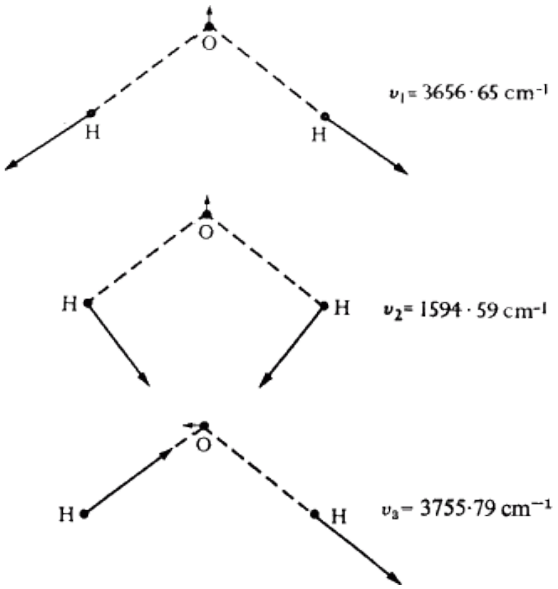


Figure 21: The normal modes of vibration of H_2O. Taken from [6, Fig. 1.1].

In this work, the vibration modes were studied under the harmonic approximation. Let us denote by $G(v_1, v_2, v_3)$ the energy above the vibrationless equilibrium state of the state with quantum numbers v_1, v_2 and v_3. Then

$$G(v_1, v_2, v_3) = \sum_{i=1}^{3} \omega_i \left(v_i + \frac{1}{2} \right) + \sum_{i=1}^{3} \sum_{k \geq i}^{3} x_{ik} \left(v_i + \frac{1}{2} \right) \left(v_k + \frac{1}{2} \right) \qquad (78)$$

where the sums are over normal modes. The ωs in this equation are often called the *harmonic frequencies*; they are the frequencies with which the molecule would vibrate if its vibrations were perfectly harmonic. The xs are the *anharmonic constants* and describe the effect on the vibrational frequencies of the departure from purely harmonic form of the vibrations. Table 3 and Table 4

contain the harmonic frequencies for H_2O and the errors for each model compared with reference data.

Model	ω_1	ω_2	ω_3
Reference	3832.17	1648.47	3942.53
small	3722.10	1485.10	3841.80
medium	3592.20	1601.10	3736.50
large	3695.60	1497.30	3814.80
MACE-ICE13-1	3702.30	1607.80	3807.50

Table 3: The harmonic frequencies of the normal modes of the water molecule, expressed in cm^{-1}. Reference data from [6, Table 1.4].

Model	ω_1	ω_2	ω_3
Reference	0.00	0.00	0.00
small	-110.07	-163.37	-100.73
medium	-239.97	-47.37	-206.03
large	-136.57	-151.17	-127.73
MACE-ICE13-1	-129.87	-40.67	-135.03

Table 4: Errors on the harmonic frequencies of the normal modes of the water molecule, expressed in cm^{-1}.

Studying the vibrational properties of the geometry obtained at the end of the optimization procedure also allows us to assess if the final geometry is a stable or unstable configuration. The vibrational modes are calculated from a finite difference approximation of the Hessian matrix, displacing atoms according to a parameter named `delta`, measured in Å.[5]

The following figures represent the frequencies of the vibration modes of the optimized water molecule obtained with ASE plotted against the displacement parameter `delta`, using different calculator models and values of the `fmax` parameter. Frequencies are indexed in ascending order, and **imaginary frequencies**, representing unstable configurations, are shown as negative values in the graphs. Inclusion of dispersion contributions in calculators leads to minimal differences in the converged configurations.

The stability of configurations is dependent in a discriminant way on the `delta` and `fmax` parameters. The analysis confirms that the value of `fmax=1e-4` yields unstable configurations, while `fmax=1e-8` is appropriate to obtain stable configurations. Moreover, the displacement `delta` shall be smaller than `1e-4` to ensure convergence of calculations.

[5] https://wiki.fysik.dtu.dk/ase/ase/vibrations/modes.html

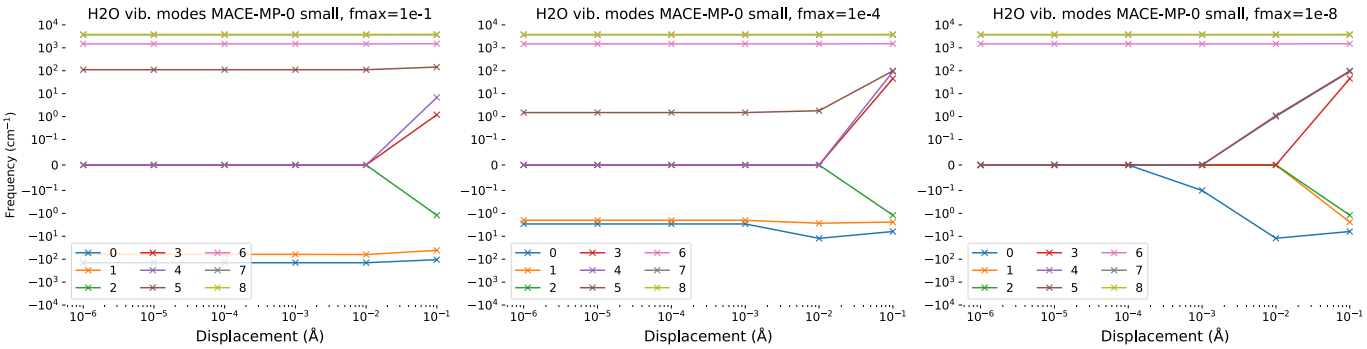


Figure 22: Convergence of the small model with respect to fmax

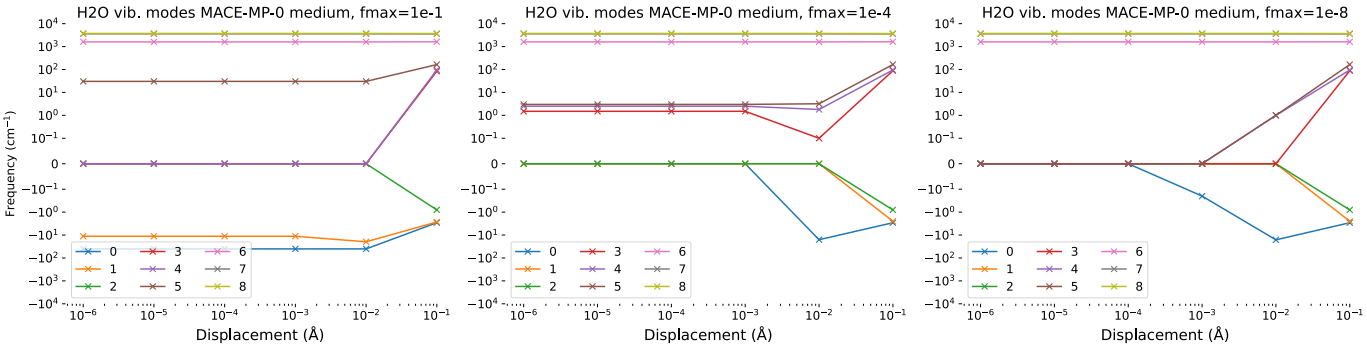


Figure 23: Convergence of the medium model with respect to fmax

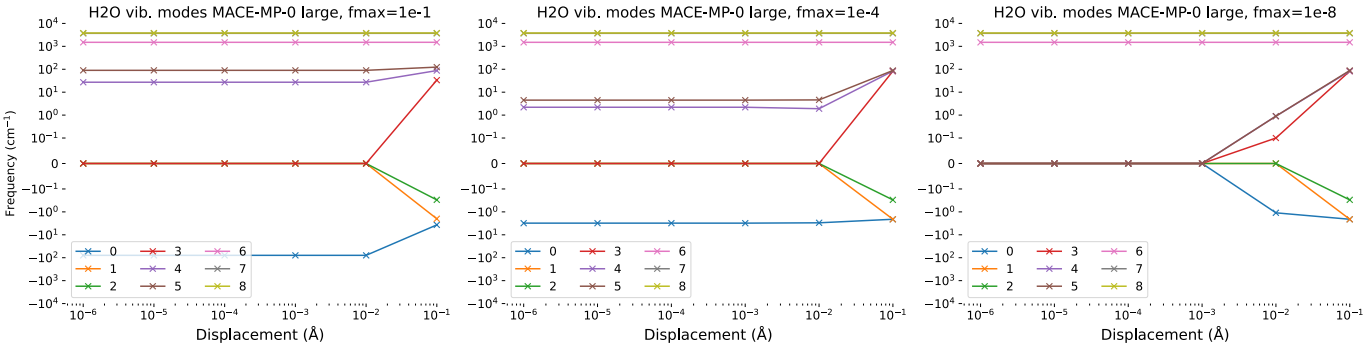


Figure 24: Convergence of the large model with respect to fmax

Adding the dispersion correction to the models in this step produces insignificant differences in the results, so their graphs are omitted for brevity.

3.1.3.1 MACE-ICE13-1

The MACE-ICE13-1 model is fine-tuned from the MACE-MP-0 medium model with dispersion.

Convergence of results for the MACE-ICE13-1 model is achieved for displacements below 10^{-4}Å. The imaginary frequency observable in Figure 25 corresponds to negligible energy, as can be observed in the representative output in Listing 2 and therefore does not pose a issue.

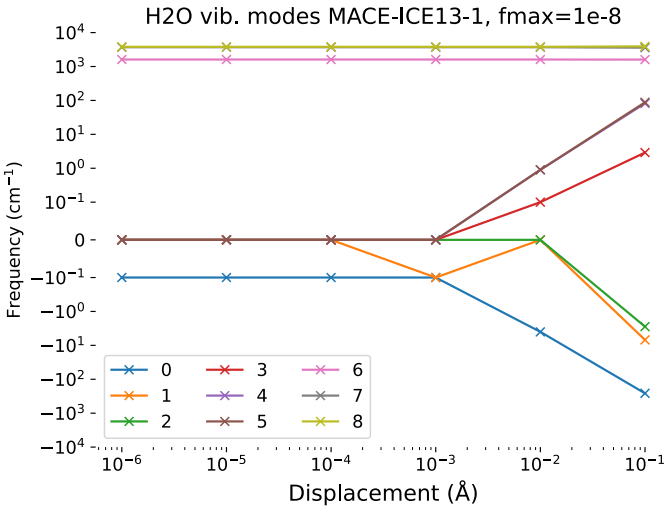


Figure 25: Frequencies calculated with the MACE-ICE13-1 model with the most restrictive `fmax`.

```
---------------------
  #    meV      cm^-1
---------------------
  0    0.0i      0.1i
  1    0.0i      0.0i
  2    0.0i      0.0i
  3     0.0       0.0
  4     0.0       0.0
  5     0.0       0.0
  6   199.3    1607.8
  7   459.0    3702.3
  8   472.1    3807.5
---------------------
Zero-point energy: 0.565 eV
```

Listing 2: Vibrational analysis representative output from ASE for the normal modes of the water molecule, computed using MACE-ICE13-1.

```python
for d in [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]:
    vib = Vibrations(
      atoms,
      delta=d,
      nfree=4,
      name=f"vib_delta={d}"
    )
    vib.run()
    vib.summary(log=f"H2O_delta={d}_summary.txt")
    vib.clean()
```

Listing 3: Computation of vibrational properties for different values of the displacement of atoms.

3.1.3.2 Zero-point vibrational energy

The ZPE from the computation models is shown in Table 5. The MACE-ICE13-1 model shows the best agreement with reference data from literature [6], with a discrepancy of 0.01 eV.

Model	ZPE (eV)	Discrepancy (eV)
small	0.561	-0.014
medium	0.554	-0.021
large	0.558	-0.017
MACE-ICE13-1	0.565	-0.01
Reference	0.575	0.0

Table 5: Zero-point energies for the employed calculators and reference value [6]. The difference between calculated and reference value is also shown in the last column. small, medium and large models refer to MACE-MP-0.

The results so far indicate that the medium, large MACE-MP-0, and MACE-ICE13-1 approximate better the properties of the water molecule, while the small MACE-MP-0 model is the worst of them in this regard.

3.2 THE WATER DIMER

After analyzing the properties of the water molecule it is time to study a slightly more elaborate system. The water dimer exposes a bigger number of geometric features and vibrational modes. Moreover, a charactheristic physical quantity, the binding energy, allows to quantitatively evaluate the performance of the models. Resources on the water dimer are abundant, on both the theory and experiment sides. [10], [17], [45], [46], [7], [47] Harmonic approximation reference values [8] were used for comparisons in Section 3.2.2.

3.2.1 Geometry optimization

The optimization was performed starting from an initial configuration approximately reproducing the geometry represented in Figure 26. The BFGS optimizer was used with the three MACE-MP-0 models and the MACE-ICE13-1 model, with a force threshold of `fmax=1e-8` (units eV/Å). The code describing the initial geometry is available in Listing 4. The results comparing values and errors of the soft, intermolecular, features of the optimized geometry with respect to literature reference are available in Table 6 and Table 7.

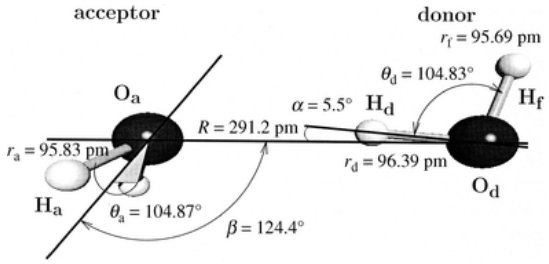


Figure 26: The equilibrium structure of the water dimer. (Image taken from [7])

```
6
# CELL(abcABC):  200.00000   200.00000   200.00000
90.00000    90.00000    90.00000  Step:          67
Bead:       0 positions{angstrom}  cell{atomic_unit}
       O  1.36346e-01 -9.67442e-01  2.40661e-01
       H -7.79751e-01 -8.69284e-01  3.06661e-02
       H  5.34721e-01 -1.44347e+00 -4.71854e-01
       O  1.42482e+00  1.38927e+00  9.26440e-01
       H  1.72033e+00  1.29319e+00  1.81419e+00
       H  9.82564e-01  5.67597e-01  7.04541e-01
```

Listing 4: The initial geometry for the optimization of the water dimer, `init.xyz`.

Model	α	θ_a	θ_d	r_{OO}(Å)	β
Reference	5.5	104.87	104.83	2.912	124.4
small	2.4	106.53	106.07	2.72	120.4
medium	6.7	104.05	105.43	2.77	99.4
large	1.6	105.29	105.23	2.82	118.0
MACE-ICE13-1	7.1	104.33	104.25	2.97	103.8

Table 6: Geometry values after optimization of the dimer, obtained from different calculator models, and from reference [7]. `small`, `medium`, `large` refer to MACE-MP-0.

Model	α	θ_a	θ_d	r_{OO}(Å)	β
small	3.1	1.66	1.24	0.19	4.0
medium	1.2	0.82	0.6	0.14	25.0
large	3.9	0.42	0.4	0.09	6.4
MACE-ICE13-1	1.6	0.54	0.58	0.06	20.6

Table 7: Errors of the geometry parameters after optimization of the dimer, obtained from different calculator models, with respect to reference values [7], [17], [18]. `small`, `medium`, `large` refer to MACE-MP-0.

The overall results exhibit a better accuracy of the MACE-MP-0 large and MACE-ICE13-1 models, holding first and second place with lowest discrepancies with respect to reference most of the time, while the small and medium MACE-MP-0 models lag behind.

A discussion on the importance of parameters can be made, considering that α is an H-bonded geometrical parameter, and can be deemed more important in the evaluation of the performance of the models, with respect to β that is not H-bonded, and is allowed a relatively bigger range of motion.

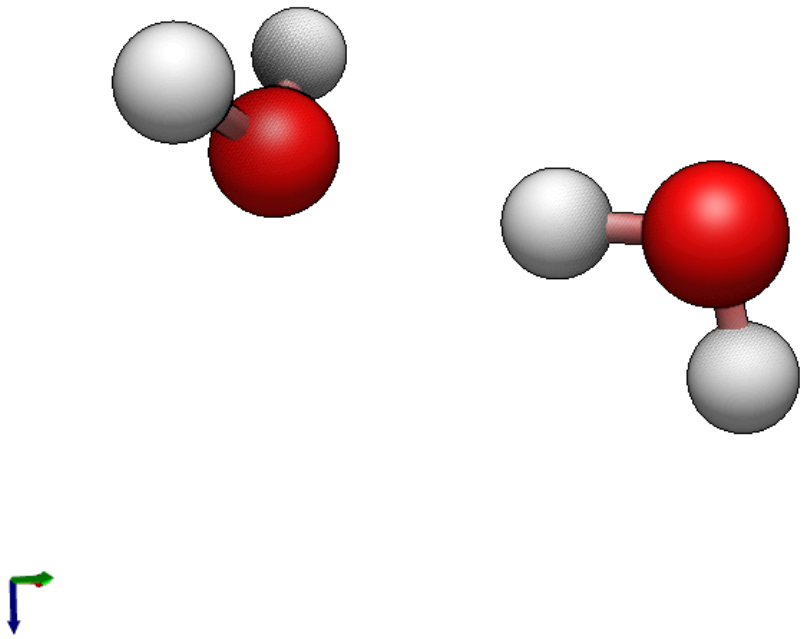


Figure 27: Render of the final geometry of the water dimer, optimized using MACE-ICE13-1.

3.2.2 Vibrations analysis

Similarly to the procedure adopted for the monomer, normal modes of the water dimer were analyzed to assess the dynamical stability of the optimized geometry. A displacement smaller than 10^{-4}Å is required for all the models to converge to stable configurations.

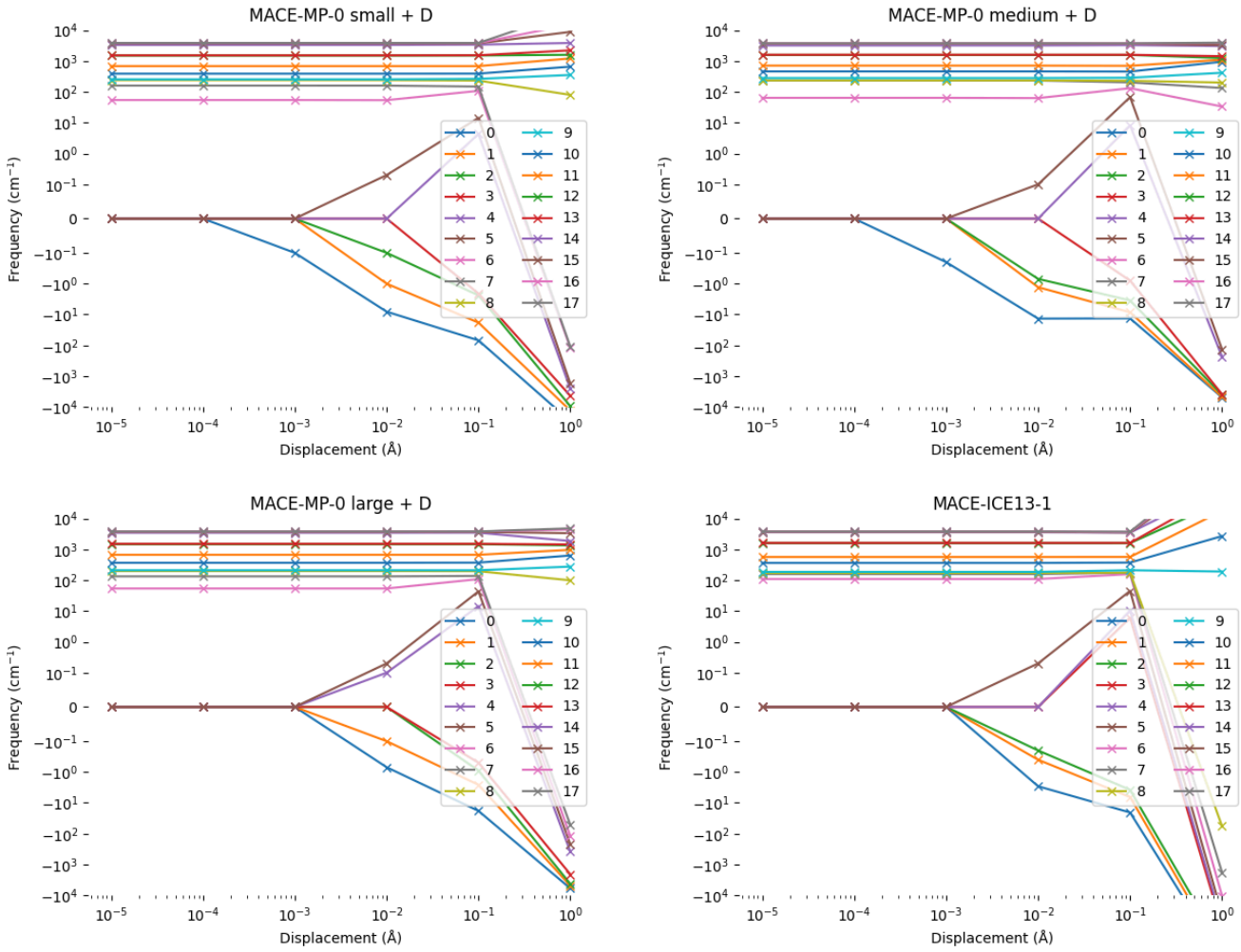


Figure 28: Structure optimization of the water dimer and vibration analysis. Negative values represent imaginary frequencies.

Table 8 and Table 9 show the comparison and the errors of the harmonic frequencies of the 12 normal modes of the water dimer with respect to reference [8]; reference values were calculated using coupled cluster theory with singlets, doublets, and perturbative triplets (CCSD(T)) with a complete basis set (CBS). Deviations from reference are also graphed in Figure 29.

	small	medium	large	MACE-ICE13-1	Reference
1	54.4	64.0	53.8	109.5	129.2
2	159.7	233.0	133.3	158.2	149.6
3	240.2	236.0	197.7	171.4	156.1
4	255.1	280.9	210.6	186.1	188.3
5	388.0	460.4	367.8	364.0	349.8
6	686.1	710.8	668.0	567.1	610.6
7	1504.2	1577.0	1473.1	1599.6	1663.0
8	1528.1	1597.8	1516.0	1622.4	1678.2
9	3331.5	3207.7	3469.6	3593.1	3754.3
10	3736.5	3635.6	3682.7	3698.6	3840.2
11	3833.3	3754.7	3781.8	3781.2	3926.8
12	3854.1	3766.1	3804.7	3800.7	3948.7

Table 8: Comparison of the harmonic normal modes frequencies obtained through MACE calculators and reference [8]. Frequency units are in cm^{-1}.

	small	medium	large	MACE-ICE13-1	Reference
1	-74.8	-65.2	-75.4	-19.7	0.0
2	10.1	83.4	-16.3	8.6	0.0
3	84.1	79.9	41.6	15.3	0.0
4	66.8	92.6	22.3	-2.2	0.0
5	38.2	110.6	18.0	14.2	0.0
6	75.5	100.2	57.4	-43.5	0.0
7	-158.8	-86.0	-189.9	-63.4	0.0
8	-150.1	-80.4	-162.2	-55.8	0.0
9	-422.8	-546.6	-284.7	-161.2	0.0
10	-103.7	-204.6	-157.5	-141.6	0.0
11	-93.5	-172.1	-145.0	-145.6	0.0
12	-94.6	-182.6	-144.0	-148.0	0.0

Table 9: Difference between the harmonic normal modes frequencies obtained through MACE calculators and reference [8]. Frequency units are in cm^{-1}.

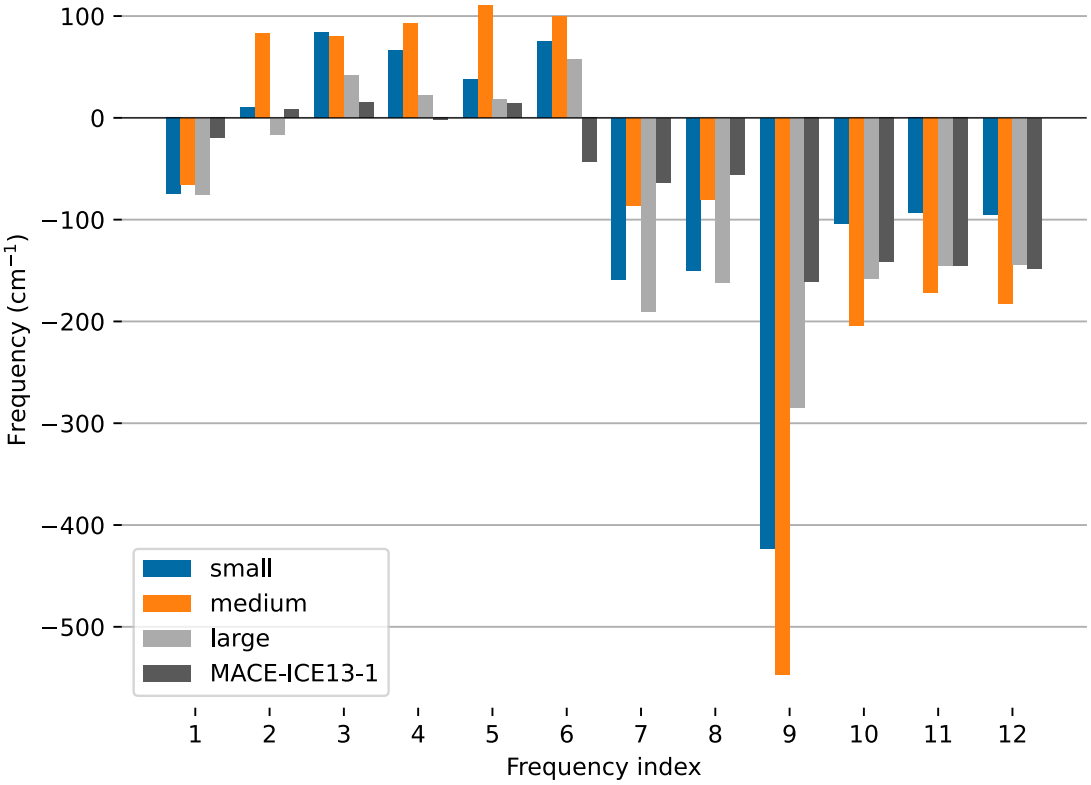


Figure 29: Deviation of the harmonic frequencies with respect to reference [8] for each MACE calculator.

In Table 10 we make a summary of the predictive power of each model of the frequencies of the normal modes of the dimer, as compared to reference [8]. MACE-ICE13-1 demonstrates the best overall adherence to the harmonic frequencies.

Model	MAE (cm^{-1})
small	114.4
medium	150.4
large	109.5
MACE-ICE13-1	68.3

Table 10: MAE computed as the difference between the value of the frequencies obtained with caluculator models and reference [8].

3.2.3 ZPE

Model	ZPE	Error
Kalescky	1.264	0.000
MACE-ICE13-1	1.218	-0.046
MACE-MP-0 small	1.213	-0.051
MACE-MP-0 medium	1.210	-0.054
MACE-MP-0 large	1.200	-0.064

Table 11: ZPE of the water dimer in the harmonic approximation and comparison with reference [8]. Energies are in units of eV.

3.2.4 Binding energy

The binding energy is calculated as

$$\Delta E_2 := E_2 - 2E_1 \tag{79}$$

where E_2 is the energy of the dimer, E_1 is the energy of one molecule.

To determine the equilibrium geometry of the dimer from the study of the binding energy, ΔE_2 must be minimized with respect to the internal coordinates. To reduce the complexity of the task, the optimization is constrained to the single degree of freedom of the $\mathrm{O}-\mathrm{O}$ distance.[6] [7] Distances between 2 and 6 Å were sampled to build the graph, with a denser sampling near the respective minima for each of the calculators. Once the two molecules are spaced apart of the fixed distance, a relaxation is performed and the potential energy of the system is calculated.

In Figure 30 the binding energy is graphed against the $\mathrm{O}-\mathrm{O}$ distance, r_{OO}, for MACE-MP-0 medium and MACE-ICE13-1. The equilibrium distance according to [9], [10], $r_{\mathrm{OO}} = 2.98\text{Å}$, is most accurately predicted by MACE-ICE13-1, while MACE-MP-0 medium is off by about a quarter of a Å.

Typical values for the binding energy obtained from DFT simulations at equilibrium position range between -20 and -12 kJ/mol (approximately -0.2 to -0.12 eV) [18], [10]. Reference [11] reports the experimental binding energy for the water dimer to be -5.44 ± 0.7 kcal/mol that corresponds to 0.24 ± 0.03 eV/particle. As MACE-ICE13-1 is trained on revPBE-D3, its predicted binding energy lies in the expected region according to DFT simulations. See Figure 31 for a representation of typical values of the binding energy obtained through DFT methods.

[6] The FixBondLength class from ASE was employed to enforce the constraint. See https://wiki.fysik.dtu.dk/ase/ase/constraints.html#the-fixbondlength-class

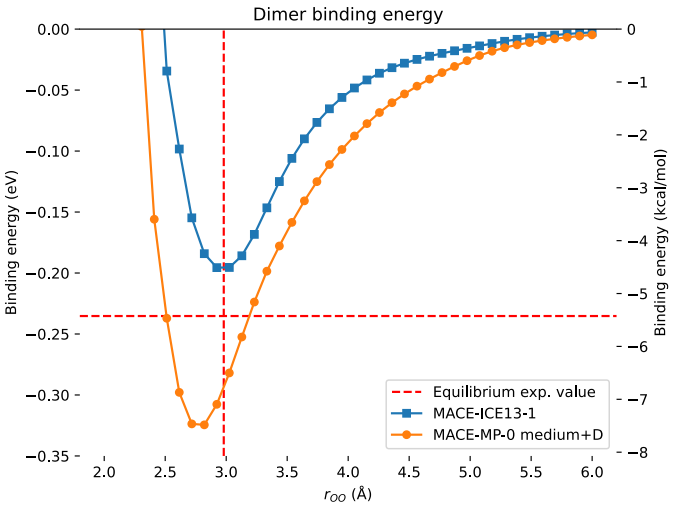


Figure 30: Dimer binding energy for different calculators. The vertical dashed line corresponds to the reference value for the equilibrium OO distance, corresponding to the experimental value of 2.98Å [9], [10]. The horizontal dashed line corresponds to −0.24 eV [11].

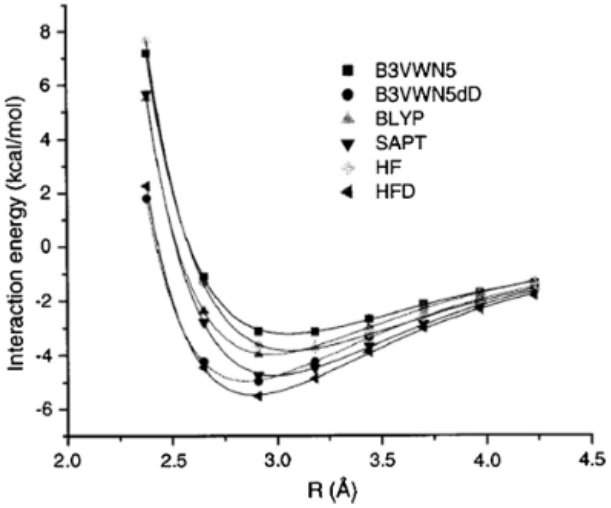


Figure 31: Image from reference [10, Fig. 5], *The interaction energy of the water dimer for the equilibrium geometry configuration calculated at different center of mass separations between the two monomers*, using DFT and other lower level methods calculations.

4 | RESULTS II: CRYSTAL STRUCTURES

This chapter presents the results of the study, focusing on the properties of molecular crystals modeled using machine learning potentials. The analyses include geometry optimization, vibrational analysis, lattice energy computations, and crystal phonons calculations.

4.1 LATTICE ENERGY

The work in this section is based on [12]. The correct estimate of the lattice energies, absolute and relative, is of great interest for water and molecular crystals in general. According to the article, the best values for the energies are consiered to be the Diffusion Monte Carlo (DMC) calculations. Regarding DFT methods, [14] found that the revPBE functional gives the closest structure and dynamical properties of ice Ih.

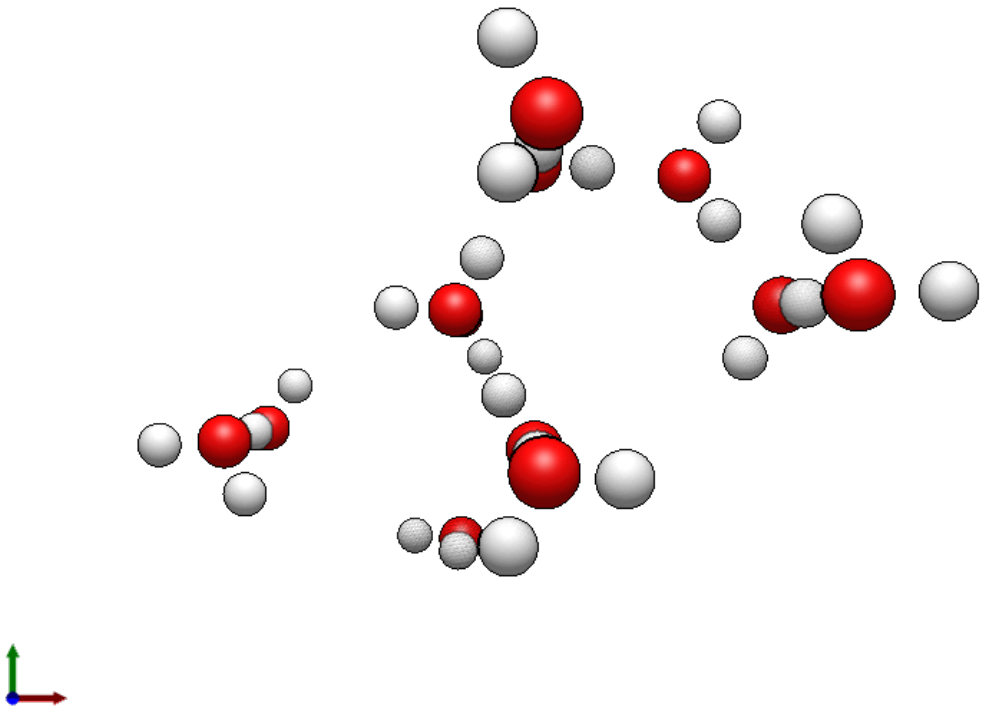


Figure 32: Render of the ice Ih cell.

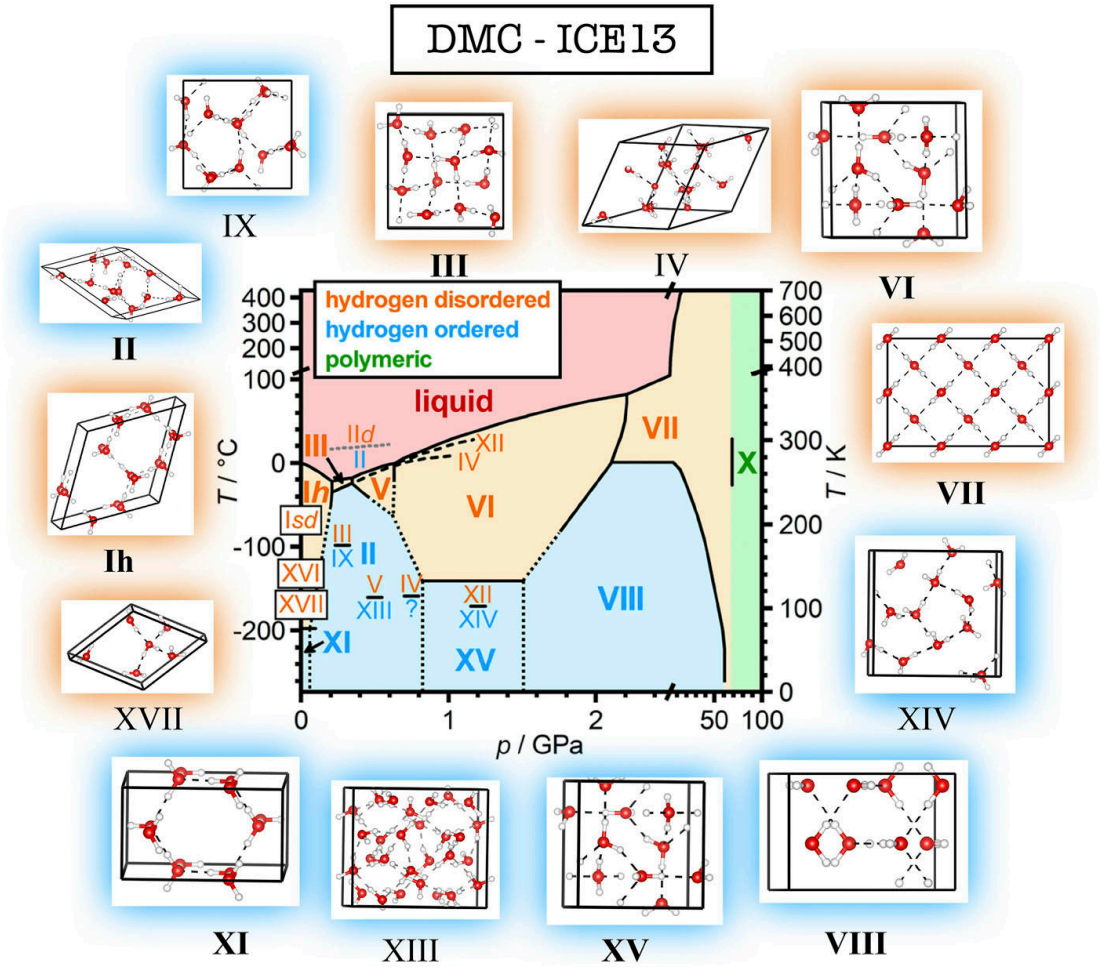


Figure 33: Crystalline structures of the systems contained in the DMC-ICE13 dataset. Image obtained from [12], [13].

4.1.1 Absolute lattice energy

The physical quantity usually considered to establish the stability of a crystal is its absolute lattice energy, which is the energy per molecule gained upon assuming the crystal form with respect to the gas phase. It can be computed as [12]

$$E_{\text{lattice}} := E_{\text{crystal}} - E_{\text{gas}}, \tag{80}$$

where E_{crystal} is the energy per molecule in the crystal phase, and E_{gas} is the energy of the isolated molecule.

$$E_{\text{crystal}} := \frac{E}{N_{\text{H}_2\text{O}}} \tag{81}$$

The quantity E_{gas} is calculated in the same manner as in Section 3.1, however, with the distinction that optimization was not performed; to align the computed

results with the reference paper, the same fixed gas phase molecule was used, referenced as Patridge 1997 from the original authors, to correctly reproduce the computational setup.

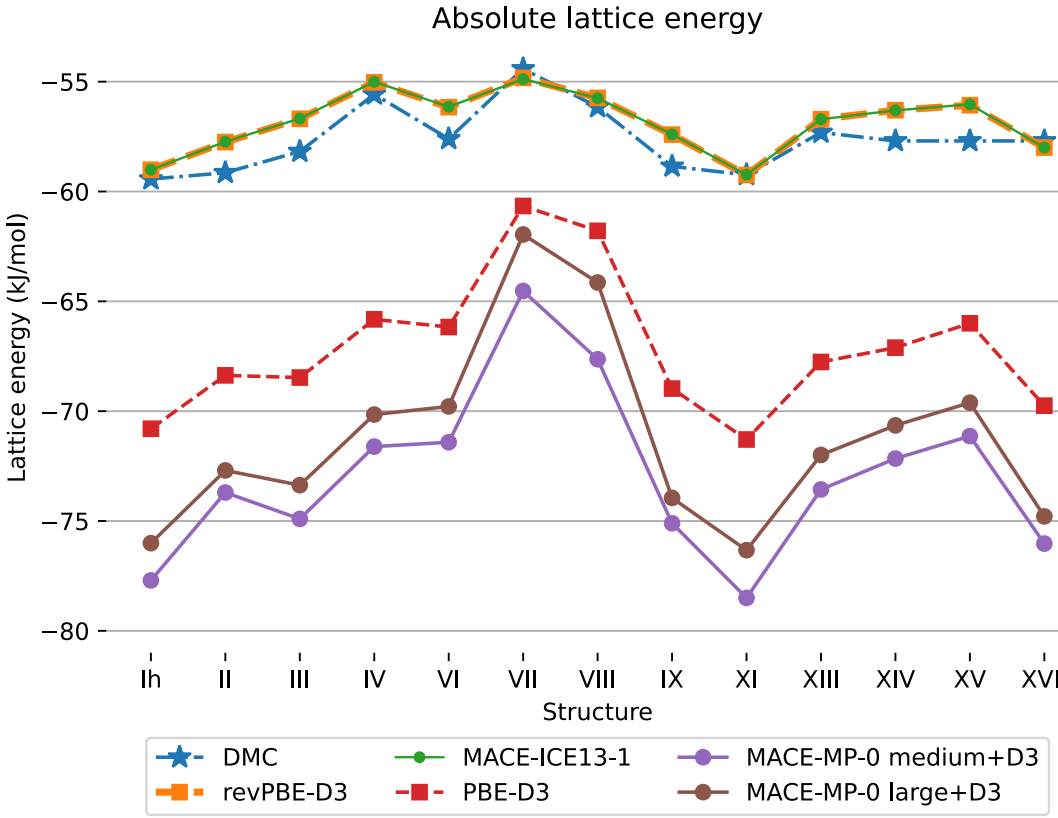


Figure 34: Performance of MACE for the 13 ice polymorphs considered on the absolute lattice energy, compared with reference models from [12].

The MACE-ICE13-1 model achieves a MAE with respect to DMC of 0.90 kJ/mol. Exchange–Correlation (XC) functionals are generally classified as good if their MAE is $\lesssim$4 kJ/mol ($\lesssim$2 kJ/mol) for the absolute (relative) lattice energy. [12, §III]

A possible improvement in the quality of calculations could be investigated employing bigger supercells for the calculation of lattice energies. We could expect a different yield for the MLPs as force field potentials are particularly suitable for dealing with a large number of particles, and the Ewald summation might be relevant for the quality of results in the present setting.

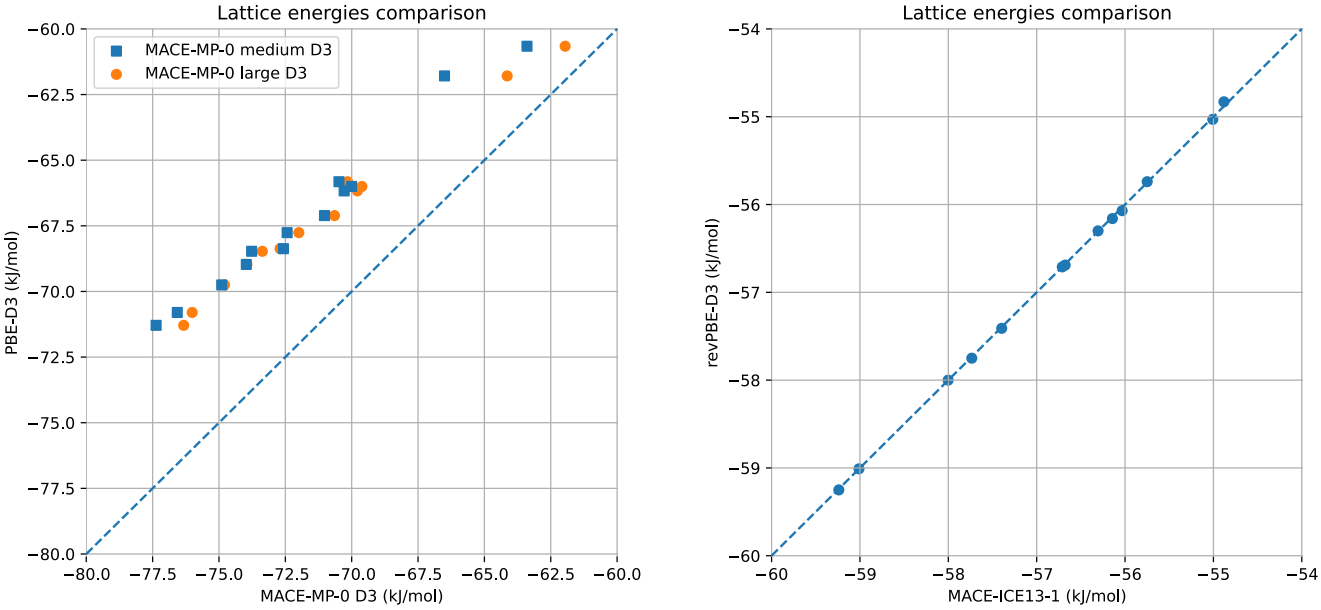


Figure 35: Scatter plot comparison of MACE models versus their respective reference models.

4.1.2 Relative lattice energy

In [12] it is noted that there is interest in capturing the relative stability of the ice polymorphs, i.e., the stability with respect to a fixed crystalline phase instead of the gas state. This property is more relevant in, e.g., the computation of the water phase diagram. Therefore, we assess the relative stability of the crystalline phases by computing the relative lattice energy with respect to hexagonal ice Ih.

For a general polymorph x, the relative lattice energy is computed as

$$\Delta E^{x}_{\text{lattice}} := E^{x}_{\text{lattice}} - E^{\text{Ih}}_{\text{lattice}} \tag{82}$$

and is independent of the configuration of the monomer in the gas phase. Figure 36 shows the computed relative lattice energy for the 13 selected polymorphs of water. MACE-ICE13-1 shows a great adherence to its reference potential, revPBE-D3, positioning itself closely to the DMC quality of results.

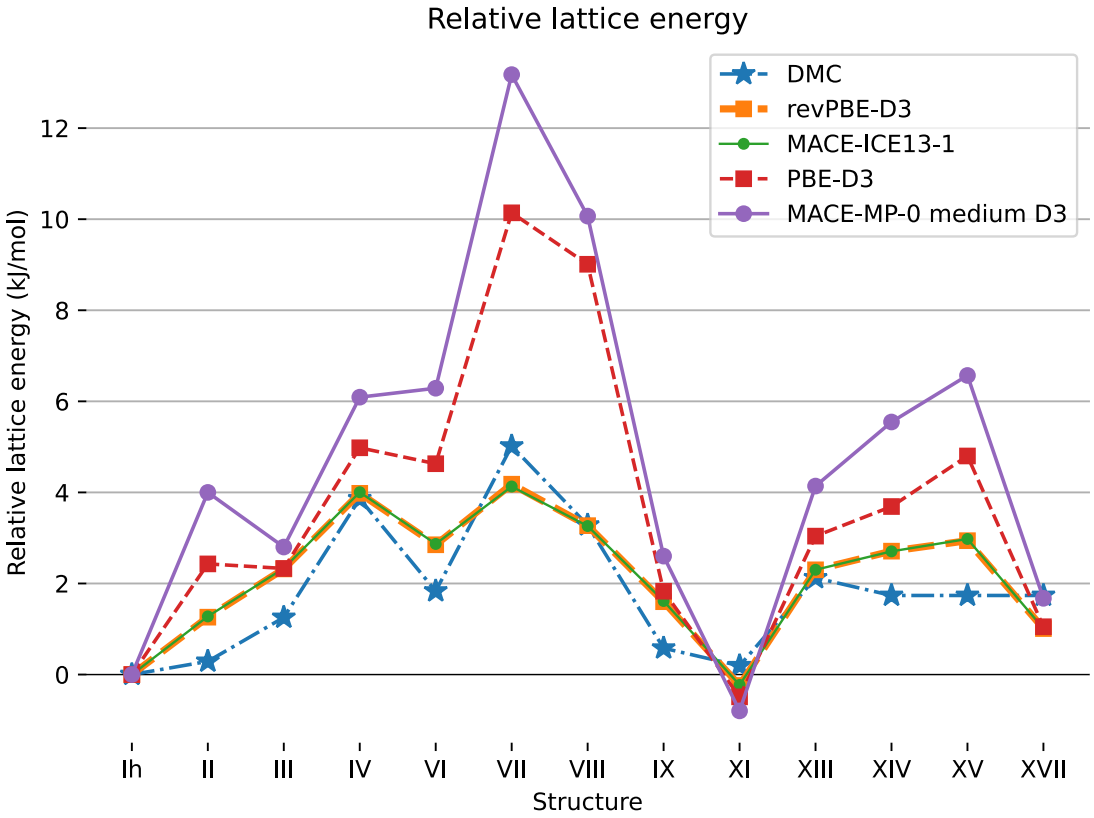


Figure 36: Performance of MACE for the 13 ice polymorphs considered on the relative lattice energy, compared with reference models. [12]

4.2 CRYSTAL PHONONS

Normal modes of vibration are calculated using the so-called **small displacement method**. This method is increasingly more accurate with bigger and bigger supercells.

For details on the tools used for phonons calculations, see Section 5.4.

The first thing to do is to build a supercell. The Ih ice structure was analyzed; it is composed of 36 atoms, that is 12 water molecules. The biggest supercell that the GPU cuda version of MACE can handle is the $3 \times 3 \times 3$. Using MACE on CPU allows us to employ also $4 \times 4 \times 4$ supercells, at the expense of a significantly increased computation time.

4.2.1 Band structure

To build the band structure of ice Ih, we have to define the band path on which the frequencies are calculated. The crystal form of ice Ih is hexagonal.[7] The Brillouin zone of the hexagonal lattice is also hexagonal. In the Brillouin zone

[7] https://en.wikipedia.org/wiki/Phases_of_ice#Known_phases

one can find several points of high symmetry that are of special interest, called *critical points*. Namely:[8]

- Γ: center of the Brillouin zone
- A: center of a hexagonal face
- H: corner point
- K: middle of an edge joining two rectangular faces
- L: middle of an edge joining a hexagonal and a rectangular face
- M: center of a rectangular face

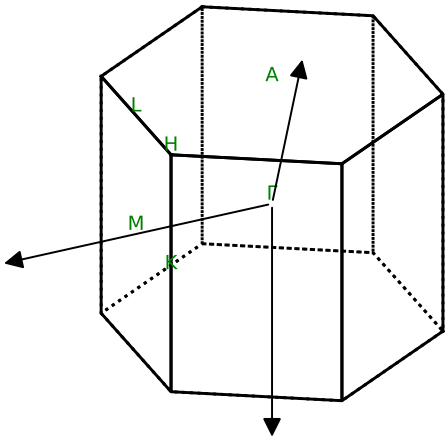


Figure 37: Special points of the Brillouin zone of ice Ih.

The bandpath is chosen following the reference article [14], that is sampling the Brillouin zone passing through critical points $\Gamma, A, K, H, M, L, \Gamma$ by means of straight line segments. See Figure 39 for the graphical representation of the bandpath. The line segments are sampled with a number of points, named q-points. The number of q-points in each path including end points is chosen as 101.

[8] https://en.wikipedia.org/wiki/Brillouin_zone#Critical_points

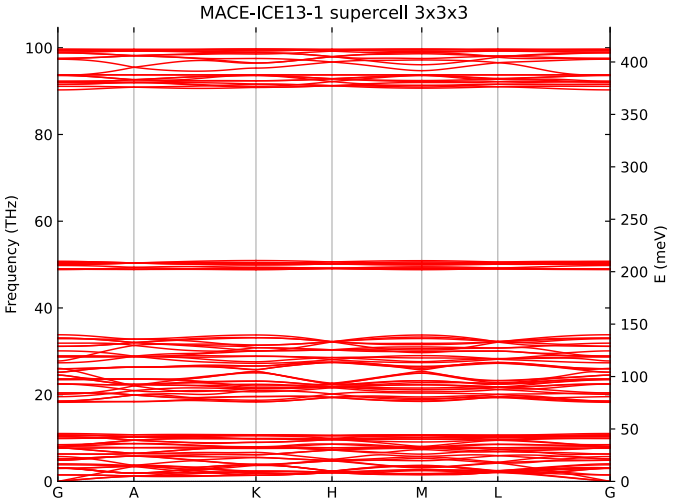


Figure 38: The bandstructure of ice Ih, computed using MACE-ICE13-1 and a $3 \times 3 \times 3$ supercell.

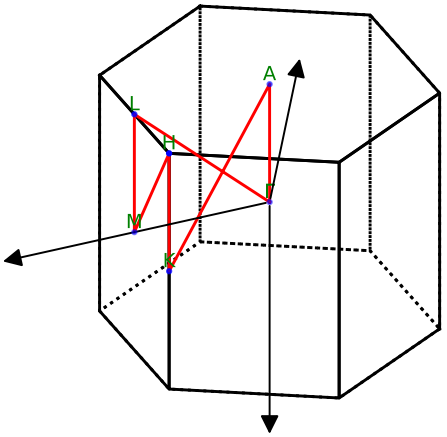


Figure 39: The bandpath chosen by [14].

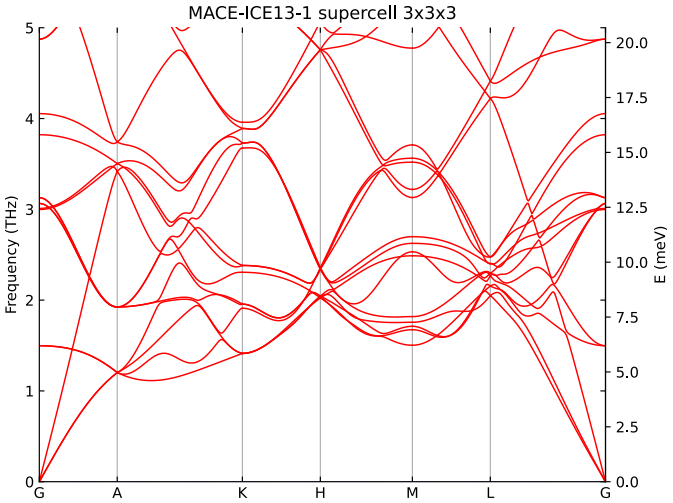


Figure 40: Phonon bandstructure of ice Ih, computed using MACE-ICE13-1.

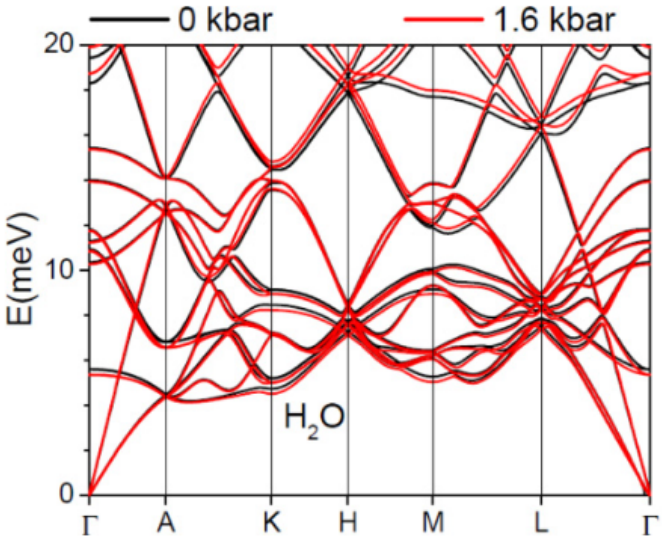


Figure 41: Phonon bandpath dispersion reference, taken from [14, Fig. 4].

The bandstructure calculations result in a approximate reproduction of reference data, as can be observed comparing Figure 40 and Figure 41. Frequencies are significantly higher than reference; the source of this discrepancy is not clear; varying volumes of the cell were tested and did not change the result; another source of the issue could be a built-in deviation of the calculator. This issue has yet to be fully investigated at the time of writing.

Calculation of phonons dispersion along the band path was also performed using the PHON code [20] for consistency of calculations. The obtained results are in accordance with reference and previous calculations, maintaining the charachteristic over-estimation of the phonon frequencies we saw before.

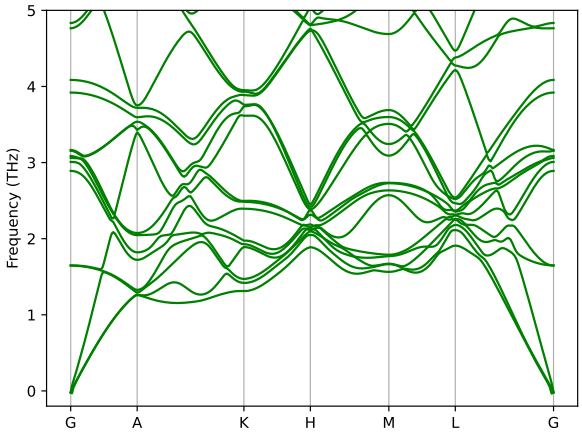


Figure 42: Phonons bandstructure of ice Ih, computed with PHON using MACE-ICE13-1.

Frequencies calculated with MACE-MP-0, shown in Figure 44, exhibit even higher frequencies and are reputed as lower quality for the current analysis. A visual qualitative analysis of the band structure produced with MACE-MP-0 exposes several different behaviours, particularly at zone boundary points; most notably, see the disalignment of bands around point M, and at point A.

As reference [14] employed a $2 \times 2 \times 2$ supercell for its calculations, we tested the convergence of calculations with respect to a higher supercell. Limitation of resources allowed the calculation of the frequencies using at most $3 \times 3 \times 3$ supercell. The comparison of the results with the different supercells is shown in Figure 43.

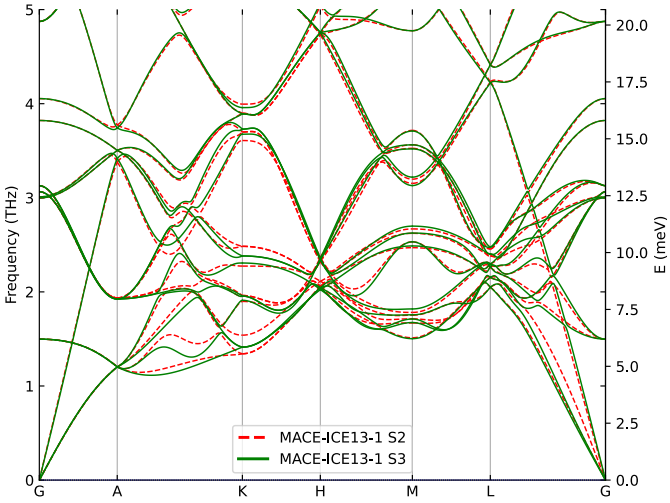


Figure 43: Comparison of the bandstructures computed with $2 \times 2 \times 2$ and $3 \times 3 \times 3$ supercells.

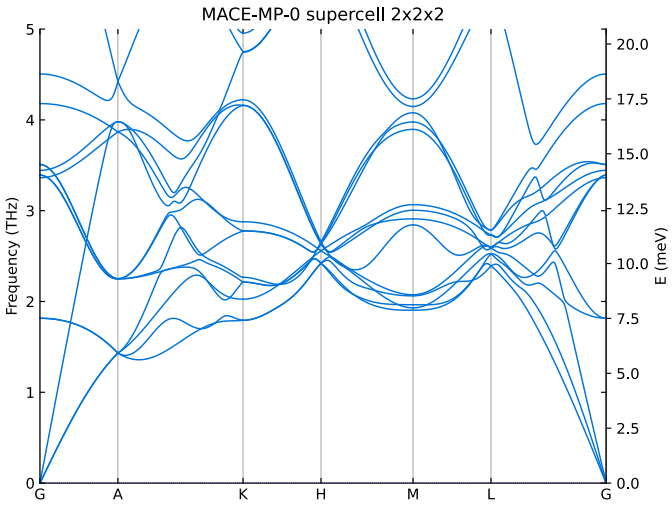


Figure 44: Bandstructure of ice Ih computed with MACE-MP-0.

4.2.1.1 Timing

supercell	device	optimization time	forces time
1	cuda	45s	8s
2	cuda	44s	50s
3	cuda	44s	21s
4	cpu	5m 23s	1h 47m 7s

Table 12: Execution times with phonopy. [19]

supercell	time	device
2	1m 30s	cuda
4	fail (out of memory)	cuda
4	7h 32m	cpu

Table 13: Execution times with Phonons by ASE.

supercell	forces time	dispersions time	device
3	3m 22s	22s	cuda

Table 14: Execution times with PHON. [20]

4.2.2 Phonons DOS

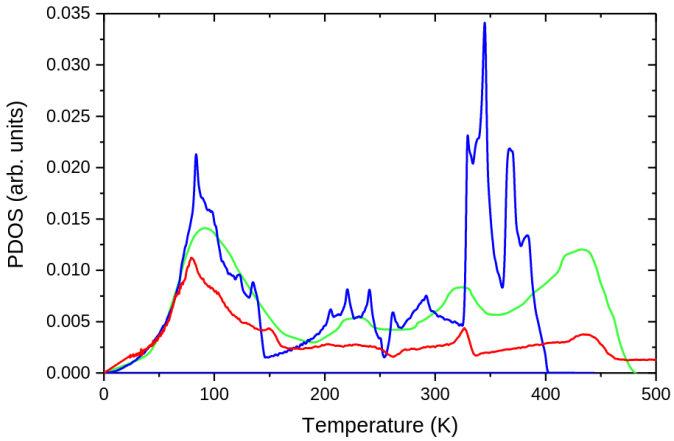


Figure 45: Reference phonons DOS, taken from [15]. Comparison of the experimental phonons DOS (green curve) with the theoretical phonons DOS (blue curve) and the neutron scattering function (red curve with roughly adjusted scale).

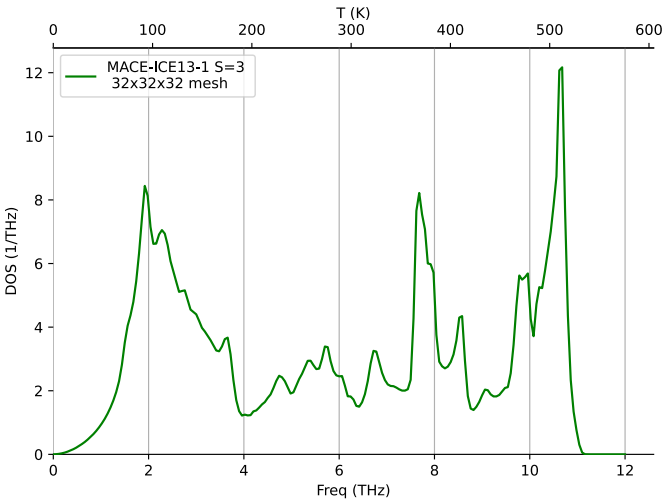
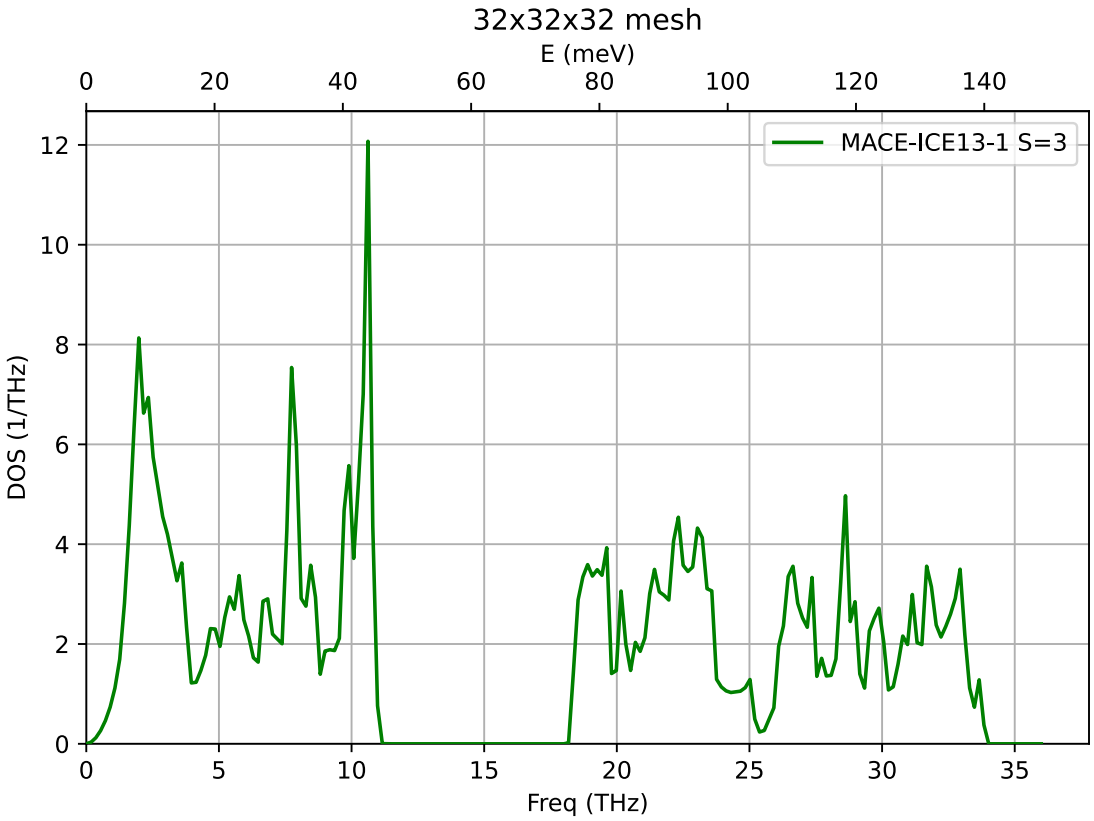


Figure 46: Calculated phonons DOS, using MACE-ICE13-1 and smearing width $\sigma = 0.05$.

As anticipated in the analysis of the bandstructure, frequencies calculated with MACE-ICE13-1 are shifted toward higher values, compared to reference data.



4.2.3 Heat capacity

[48] treats the heat capacity of ice at low temperatures. [15] provides an all-round thermodynamic model of ice Ih, detailing the analysis of heat capacity with 1 Debye and 7 Einstein terms; the analysis of heat capacity considering harmonic terms is reproduced along with reference data in Figure 47.

In the quasi-harmonic approximation the lattice vibrations are assumed to be harmonic but with frequencies dependent upon the volume. [49]

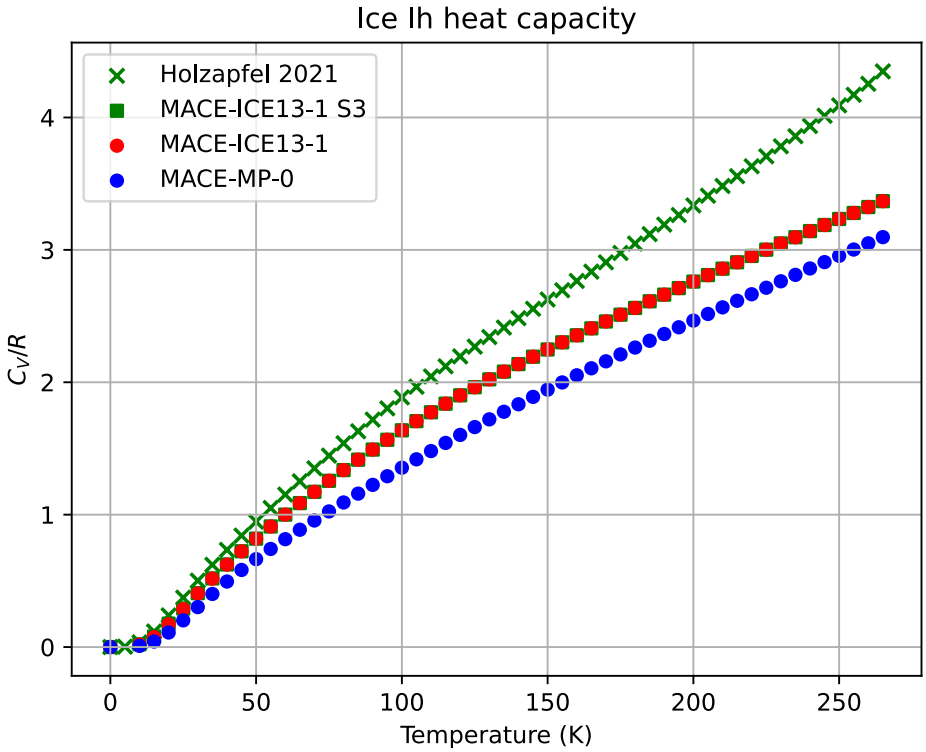


Figure 47: Heat capacity of ice Ih. Comparison of results from simulation with different calculators (S3 indicates supercell 3x3x3, otherwise supercell is 2x2x2) and reference data [15].

4.2.4 Band structure and DOS of D_2O

A further study was performed to analyze the performance of the calculator compared with reference data on deuterated water. [16] In this scenario, the fidelity is estimated to be worse than the previous case. The same considerations on the quality of the results hold as above.

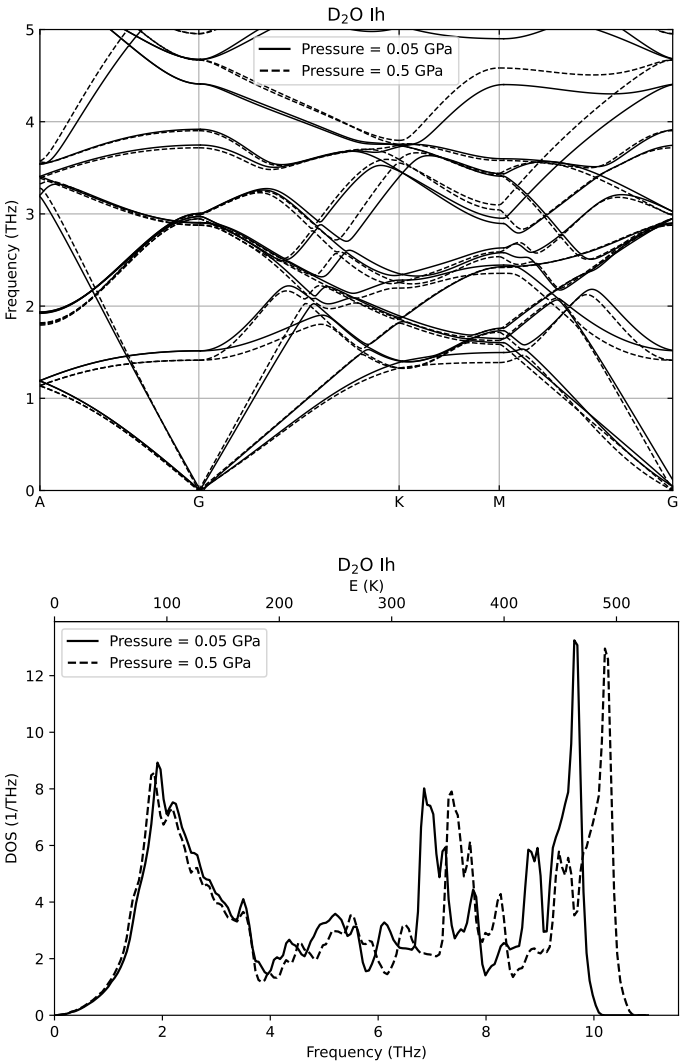


Figure 48: Band structure and DOS calculated with MACE-ICE13-1.

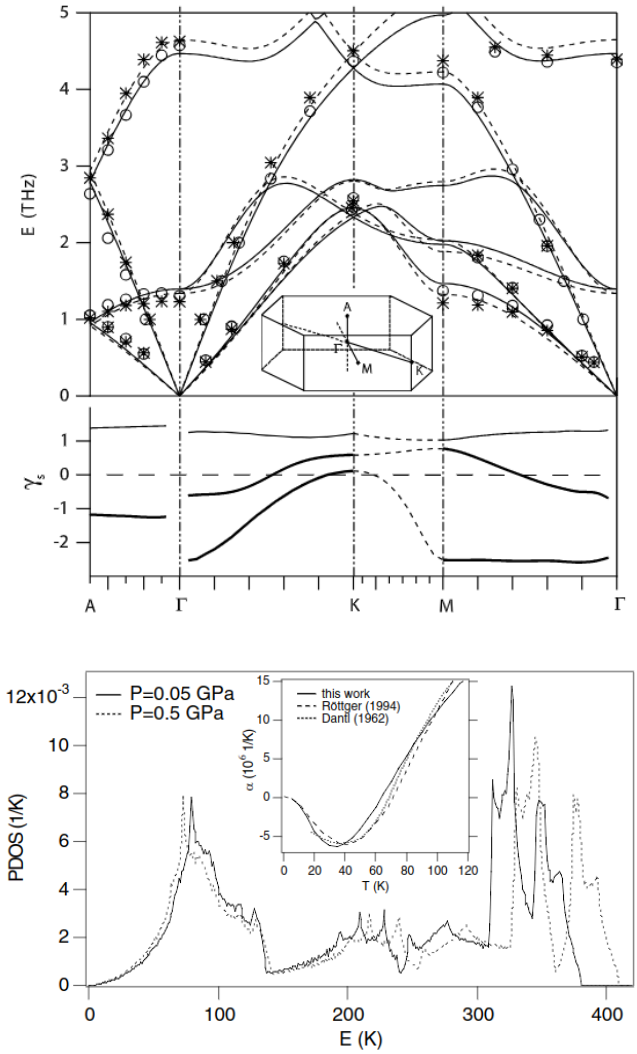


Figure 49: Band structure and DOS from reference [16].

4.3 MD

4.3.1 RDF

Constant NVT MD simulations with Langevin thermostat[9] were performed under varying external conditions. A thermostat couples the system to an external heath bath. The Radial Distribution Function (RDF) of the thermalized states is shown in Figure 50, compared with reference data [50] from X-ray diffraction experiment. The simulated phisical time shall not be less than 100ps, to allow recombination of bonds in the liquid. Constant NPT simulations should be more appropriate for the computation of physical properties, but they are

[9] https://wiki.fysik.dtu.dk/ase/ase/md.html#module-ase.md.langevin

missing at the present time. Further analysis can also be made on the study of the diffusion coefficient and the density of the system.

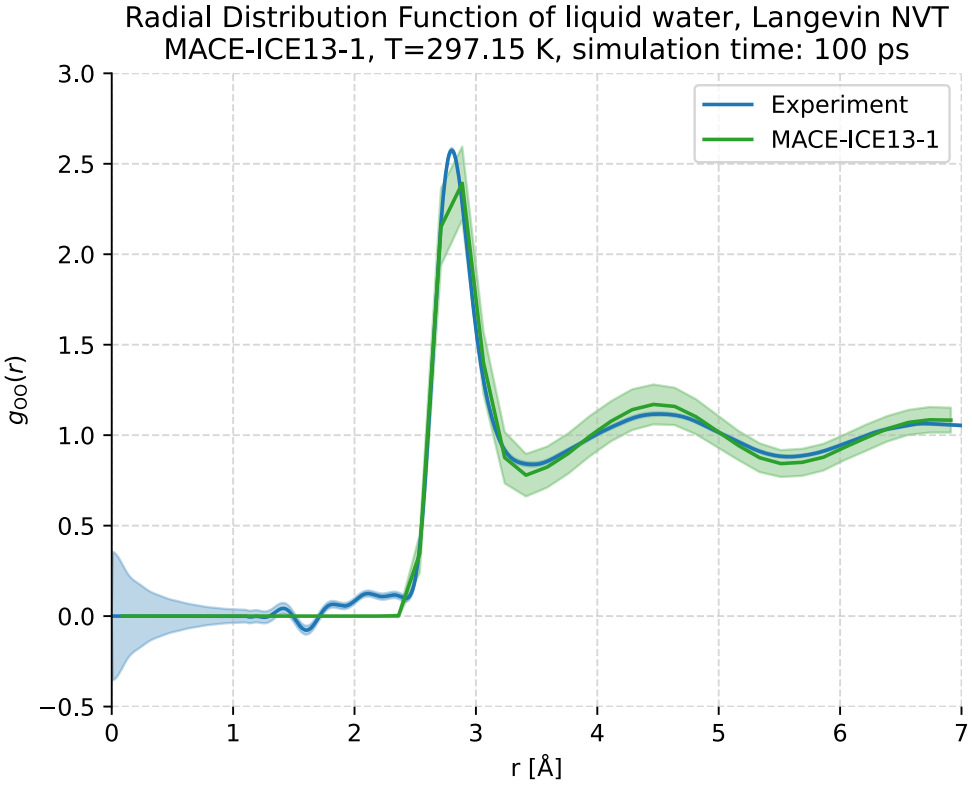


Figure 50: Radial distribution function of oxygens in liquid water.

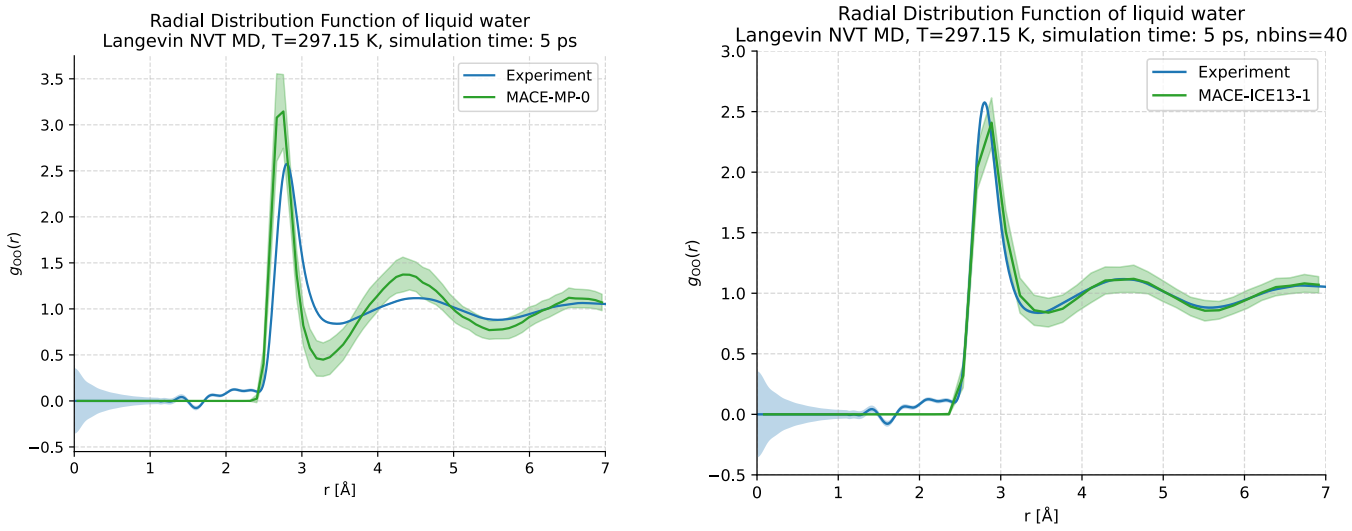
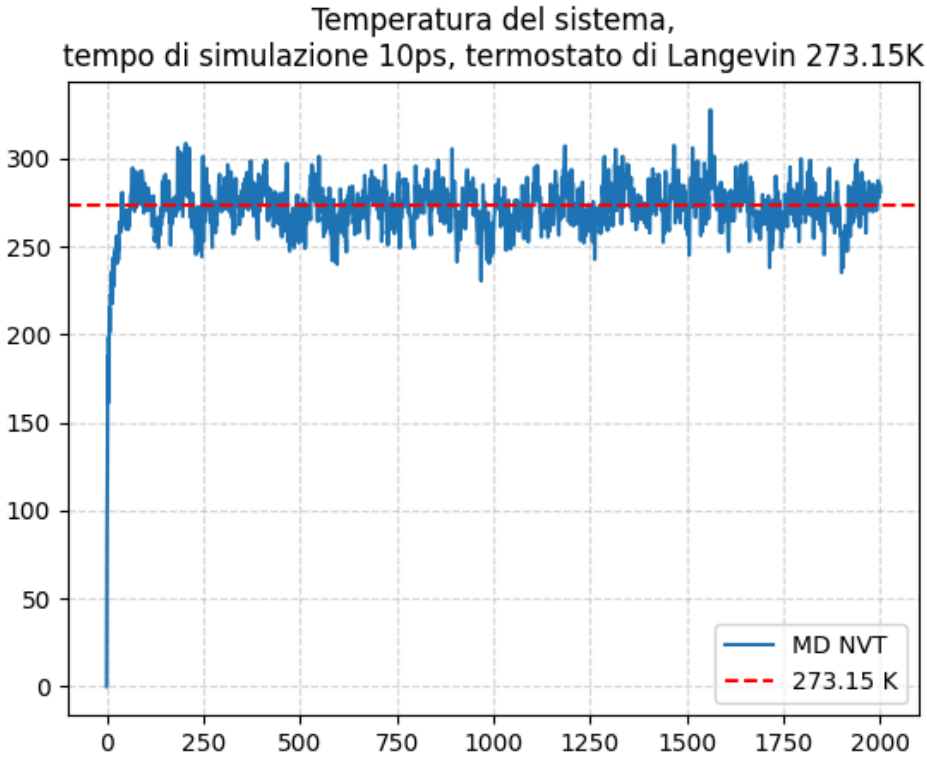


Figure 51: Comparison of the RDFs obtained from MD simulations of liquid water using MACE-MP-0 and MACE-ICE13-1.


Temperatura del sistema,
tempo di simulazione 10ps, termostato di Langevin 273.15K
300
250
200
150
100
50
0
0
250
500
750
1000
1250
1500
1750
2000
MD NVT
273.15 K

5 | TOOLS

5.1 HARDWARE: IBISCO

Simulations were performed on the Infrastructure for BIg data and Scientific Computing (IBiSCo) cluster provided by the Federico II University. [51] The architecture of the hybrid cluster of the IBiSCo Data Center can be represented as a set of multiple layers. The lowest layer of the architecture consists of the hardware, characterized by calculation and storage nodes; in the upper level the application level, which allows users to submit their tasks. The intermediate level of the architecture consists of the set of CUDA and MPI libraries which are capable of making the two levels communicate with each other.

5.1.1 The hardware level

The cluster comprises 36 nodes and 2 switches, placed in 4 racks of the Data Center. They perform two functions: calculation and storage. To support the calculation there are 128 GPUs, distributed among 32 nodes (4 GPUs per node). To support storage, 320 TB are available distributed among 4 nodes (80 TB per node). To ensure access to resources and low-latency broadband communication between nodes, the InfiniBand technology is used to provide a high-performance network.

5.1.2 The Compute Node Architecture

The cluster compute nodes are 32 Dell C4140s, each equipped with 4 NVIDIA V100 GPUs, 2 Ethernet ports at 10Gb/s each, 2 InfiniBand ports at 100Gb/s each, 2 Intel Gen 2 Xeon Gold CPUs, and 2 SATA 480 GB SSDs. Each node is also equipped with 22 64 GB RAM memory modules, overall 1.375 TiB. Each GPU is equipped with 34 GB RAM memory. The nodes are divided into 3 differently sized sub-clusters.

5.2 FRAMEWORK: ASE

The Atomic Simulation Environment (ASE) is a set of tools and Python modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations. ASE provides interfaces to different codes through `Calculators` which are used together with the central `Atoms` object and the many available algorithms in ASE. The `Atoms` object contains the positions of the atoms and the properties of the cell. ASE allows geometry optimization, computing of the

potential energy of the system, molecular dynamics. The MACE code was executed through the calculators interface of ASE.

5.2.1 Structure optimization

The optimization algorithms can be roughly divided into local optimization algorithms which find a nearby local minimum and global optimization algorithms that try to find the global minimum (a much harder task). Most optimization algorithms available in ASE accept the same base parameters:

- The atoms object to relax
- A log file
- An attached trajectory object

Basic optimizers optimize only internal atomic positions. Cell volume and shape can also be optimized in combination with Filter tools.

The local optimization algorithms available in ASE follow the convergence criterion, which asks that the force on all individual atoms should be less than *fmax*:

$$\max_a |\vec{F}_a| < f_{\max}. \tag{83}$$

The BFGS algorithm uses two quantities to decide where to move the atoms on each step:

- the forces on each atom, as returned by the associated calculator;
- the Hessian matrix, i.e. the matrix of second derivatives $\frac{\partial^2 E}{\partial x_i \partial x_j}$ of the total energy with respect to nuclear coordinates.

If the atoms are close to the minimum, such that the potential energy surface is locally quadratic, the Hessian and forces accurately determine the required step to reach the optimal structure. The Hessian is very expensive to calculate a priori, so instead the algorithm estimates it by means of an initial guess which is adjusted along the way depending on the information obtained on each step of the structure optimization.

5.3 CALCULATOR: MACE

MACE [52], [53] is an equivariant message-passing graph tensor network where each layer encodes many-body information of atomic geometry. At each layer, many-body messages are formed using a linear combination of a tensor product basis. [54], [55] This is constructed by taking tensor products of a sum of two-body permutation-invariant polynomials, expanded in a spherical basis. **The final output is the energy contribution of each atom to the total potential energy.** The MACE architecture is implemented in PyTorch and employs the e3nn library.

In the following sections the technical details of the internals of MACE will be exposed. A brief introduction to the topics that follow, MPNNs and GNNs, is detailed in Section 2.6.

5.3.1 MPNN Interatomic Potentials

Message Passing Neural Networks (MPNNs) are a type of Graph Neural Network (GNN) that parametrizes a mapping from a labeled graph to a target space, either a graph or a vector space. [56] When applied to parametrize properties of atomistic structures (materials or molecules), the graph is embedded in 3-dimensional (3D) Euclidean space, where each node represents an atom, and edges connect nodes if the corresponding atoms are within a given distance of each other. The state of each node i in layer t of the MPNN is represented by a tuple

$$\sigma_i^{(t)} = \left(\vec{r}_i, z_i, \vec{h}_i^{(t)}\right), \tag{84}$$

where $\vec{r}_i \in \mathbb{R}^3$ is the position of atom i; z_i is the chemical element; $\vec{h}_i^{(t)}$ are its learnable features. A forward pass of the network consists of multiple *message construction, update* and *readout* steps. During message construction, a message $\vec{m}_i^{(t)}$ is created for each node by pooling over its neighbours:

$$\vec{m}_i^{(t)} = \bigoplus_{j \in \mathcal{N}(i)} M_t\left(\sigma_i^{(t)}, \sigma_j^{(t)}\right), \tag{85}$$

where M_t is a learnable message function and $\bigoplus_{j\in\mathcal{N}(i)}$ is a learnable, permutation invariant pooling operation over the neighbours of atom i (e.g., a sum). In the update step, the message $\vec{m}_i^{(t)}$ is transformed into new features

$$\vec{h}_i^{(t+1)} = U_t\left(\sigma_i^{(t)}, \vec{m}_i^{(t)}\right), \tag{86}$$

where U_t is a learnable update function. After T message construction and update steps, the learnable readouts functions $\mathcal{R}_t$ map the node states $\sigma_i^{(t)}$ to the target, in this case the site energy of atom i,

$$E_i = \sum_{t=1}^{T} \mathcal{R}_t\left(\sigma_i^{(t)}\right). \tag{87}$$

5.3.2 Equivariant Graph Neural Networks

In *equivariant* GNNs, internal features $\vec{h}_i^{(t)}$ transform in a specified way under some group action. [56] When modelling the potential energy of an atomic structure, the group of interest is $O(3)$, specifying rotations and reflections of the particles; translation invariance is trivially incorporated through the use of

relative distances. A GNN is called $O(3)$ equivariant if it has internal features that transform under the rotation $Q \in O(3)$ as

$$\vec{h}_i^{(t)}(Q \cdot (\vec{r}_1, ..., \vec{r}_N)) = D(Q)\vec{h}_i^{(t)}(\vec{r}_1, ..., \vec{r}_N), \tag{88}$$

where $D^{L(Q)} \in \mathbb{R}^{(2L+1)\times(2L+1)}$ is a Wigner D-matrix of order L. A feature labelled with $L = 0$ describes an invariant scalar. Features labelled with $L > 0$ describe equivariant features, formally corresponding to equivariant vectors, matrices or higher order tensors. The features of *invariant* models, such as SchNet and DimeNet, transform according to $D(Q) = \mathbb{1}$, the identity matrix. Models such as NequIP, equivariant transformer, PaiNN, or SEGNNs, in addition to invariant scalars, employ equivariant internal features that transform like vectors or tensors.

5.3.3 The MACE Architecture

The MACE model follows the general framework of MPNNs outlined in Section 5.3.1. The key innovation is a new message construction mechanism. The messages $\vec{m}_i^{(t)}$ are expanded in a hierarchical body order expansion,

$$\begin{aligned}\vec{m}_i^{(t)} = &\sum_j \vec{u}_1\left(\sigma_i^{(t)}; \sigma_j^{(t)}\right) \\ &+ \sum_{j_1,j_2} \vec{u}_2\left(\sigma_i^{(t)}; \sigma_{j_1}^{(t)}; \sigma_{j_2}^{(t)}\right) + ... + \sum_{j_1,...,j_\nu} \vec{u}_\nu\left(\sigma_i^{(t)}; \sigma_{j_1}^{(t)}; ...; \sigma_{j_\nu}^{(t)}\right),\end{aligned} \tag{89}$$

where the $\vec{u}$ functions are learnable, the sums run over the neighbours of i, and ν is a hyper-parameter corresponding to the maximum correlation order, the body order minus 1, of the message function with respect to the states.

5.3.3.1 Message Construction

At each iteration, MACE first embeds the edges using a learnable radial basis $R^{(t)}_{kl_1l_2l_3}$, a set of spherical harmonics $Y^{m_1}_{l_1}$, and a learnable embedding of the previous node features $h^{(t)}_{j,\tilde{k}l_2m_2}$ using weights $W^{(t)}_{k\tilde{k}l_2}$. [56] The $A^{(t)}_i$ features are obtained by pooling over the neighbours $\mathcal{N}(i)$ to obtain permutation invariant 2-body features whilst, crucially, retaining full directional information, and thus, full information about the atomic environment:

$$\begin{aligned}A^{(t)}_{i,kl_3m_3} = &\sum_{l_1m_1,l_2m_2} C^{l_3m_3}_{l_1m_1,l_2m_2} \cdot \\ \cdot &\sum_{j\in\mathcal{N}(i)} R^{(t)}_{kl_1l_2l_3}(r_{ji}) Y^{m_1}_{l_1}(\hat{r}_{ji}) \sum_{\tilde{k}} W^{(t)}_{k\tilde{k}l_2} h^{(t)}_{j,\tilde{k}l_2m_2},\end{aligned} \tag{90}$$

where $C^{l_3m_3}_{l_1m_1,l_2m_2}$ are the standard Clebsh-Gordan coefficients ensuring that $A^{(t)}_{i,kl_3m_3}$ maintain the correct equivariance; r_{ji} is the (scalar) interatomic dis-

tance, and $\hat{r}_{ji}$ is the corresponding unit vector. $R^{(t)}_{kl_1l_2l_3}$ is obtained by feeding a set of radial features that embed the radial distance r_{ji} using Bessel functions multiplied by a smooth polynomial cutoff to a multi-layer perceptron. In the first layer, the node features $h^{(t)}_j$ correspond to the (invariant) chemical element z_j. Therefore, Equation 90 can be further simplified:

$$A^{(1)}_{i,kl_1m_1} = \sum_{j\in\mathcal{N}(i)} R^{(1)}_{kl_1}(r_{ji})Y^{m_1}_{l_1}(\hat{r}_{ji})W^{(1)}_{kz_j}. \tag{91}$$

This simplified operation is much cheaper, making the computational cost of the first layer low.

The *key* operation of MACE is the efficient construction of higher order features from the $A^{(t)}_i$-features. This is achieved by first forming tensor products of the features, and then symmetrising:

$$B^{(t)}_{i,\eta_\nu kLM} = \sum_{\overrightarrow{lm}} \mathcal{C}^{LM}_{\eta_\nu,\overrightarrow{lm}} \prod_{\xi=1}^{\nu} \sum_{\tilde{k}} w^{(t)}_{k\tilde{k}l_\xi} A^{(t)}_{i,\tilde{k}l_\xi m_\xi}, \quad \overrightarrow{lm} = (l_1m_1, ..., l_\nu m_\nu) \tag{92}$$

where the coupling coefficients $\mathcal{C}^{LM}_{\eta_\nu}$ corresponding to the generalized Clebsh-Gordan coefficients ensuring that $B^{(t)}_{i,\eta_\nu kLM}$ are L-equivariant, the weights $w^{(t)}_{k\tilde{k}l_\xi}$ are mixing the channels (k) of $A^{(t)}_i$, and ν is a given correlation order. $\mathcal{C}^{LM}_{\eta_\nu,\overrightarrow{lm}}$ is very sparse and can be pre-computed such that Equation 92 can be evaluated efficiently. The additional index η_ν simply enumerates all possible couplings of $l_1, ..., l_\nu$ features that yield the selected equivariance specified by the L index. The $B^{(t)}_i$-features are constructed up to some maximum ν. This variable in Equation 92 is the order of the tensor product, and hence, can be identified as the order of the many-body expansion terms in Equation 89. The computationally expensive multi-dimensional sums over all triplets, quadruplets, etc…, are thus circumvented and absorbed into Equation 91 and Equation 90.

The message $m^{(t)}_i$ can now be written as a linear expansion

$$m^{(t)}_{i,kLM} = \sum_{\nu}\sum_{\eta_\nu} W^{(t)}_{z_ikL,\eta_\nu} B^{(t)}_{i,\eta_\nu kLM}, \tag{93}$$

where $W^{(t)}_{z_ikL,\eta_\nu}$ is a learnable weight matrix that depends on the chemical element z_i of the receiving atom and message symmetry L. Thus, MACE implicitly constructs each term u in Equation 89 by a linear combination of $B^{(t)}_{i,\eta_\nu kLM}$ features of the corresponding body order.

Under mild conditions on the two-body bases $A^{(t)}_i$, the higher order features $B^{(t)}_{i,\eta_\nu kLM}$ can be interpreted as a *complete basis* of many-body interactions, which can be computed at a cost comparable to pairwise interactions. Because

of this, the expansion Equation 93 is *systematic*. It can in principle be converged to represent any smooth $(\nu+1)$-body equivariant mapping in the limit of infinitely many features. [57]

5.3.3.2 Update

In MACE, the update is a linear function of the message and the residual connection: [56]

$$h_{i,kLM}^{(t+1)} = U_t^{kL}\left(\sigma_i^{(t)}, m_i^{(t)}\right) = \sum_{\tilde{k}} W_{kL,\tilde{k}}^{(t)} m_{i,\tilde{k}LM} + \sum_{\tilde{k}} W_{z_i kL,\tilde{k}}^{(t)} h_{i,\tilde{k}LM}^{(t)}. \quad (94)$$

5.3.3.3 Readout

In the readout phase, the invariant part of the node features is mapped to a hierarchical decomposition of site energies via readout functions: [56]

$$\begin{aligned} E_i &= E_i^{(0)} + E_i^{(1)} + ... + E_i^{(T)}, \quad \text{where} \\ E_i^{(t)} = \mathcal{R}_t\left(h_i^{(t)}\right) &= \begin{cases} \sum_{\tilde{k}} W_{\text{readout},\tilde{k}}^{(t)} h_{i,\tilde{k}00}^{(t)} & \text{if } t < T \\ \text{MLP}_{\text{readout}}^{(t)}\left(\left\{h_{i,k00}^{(t)}\right\}_k\right) & \text{if } t = T \end{cases} \end{aligned} \quad (95)$$

The readout only depends on the invariant features $h_{i,k00}^{(t)}$ to ensure that the site energy contributions $E_i^{(t)}$ are invariant as well. To maintain body ordering, MACE uses linear readout functions for all layers except the last, where it uses a one-layer multi-layer perceptron.

5.3.4 Performance of MACE

[58] shows that MACE generally outperforms alternatives for a wide range of systems, including liquid water. In those simulations, the many-body equivariant MACE model is an improvement with respect to the three-body atom-centered symmetry function-based feed-forward neural network model (BPNN), the three-body invariant message passing model REANN and the two-body equivariant message passing model NequIP.

MACE-MP-0 is a general-purpose Machine Learning (ML) model, trained on a public database of 150k inorganic crystals, that is capable of running stable molecular dynamics on molecules and materials. [26] The model can be applied out of the box and as a starting or "foundation model" for any atomistic system of interest and is thus a step towards democratising the revolution of ML force fields by lowering the barriers to entry. [26]

5.3.5 Fine-tuning a custom model

In this thesis, we studied and followed the procedure to create a custom made MACE model. Current developments are undergoing in the field, experimenting on the various techniques to fine-tune custom MACE models. [59] Fine-tuning a MACE model implies starting from a so called "foundation model" of MACE, e.g. MACE-MP-0, and execute further training of the model on new data. That data is comprised of desired geometries (preferably under specific thermodynamic conditions), and attached potential energies, forces and stresses of the system, computed with the experimenter's calculator of choice. The fine-tuning thus is the procedure that allows MACE to accurately reproduce the charachteristic behaviour of the chosen calculator through a new training procedure, resulting in a more informed MLP model. For the present purposes, we chose a toy model to represent ice Ih at a pressure of 1.01325 bar and a temperature of 100.0 K.

Fine-tuning a MACE model is composed of three steps, detailed as follows:

1. **Sample the phase space** to obtain representatives of the system under the thermodynamic conditions we are interesed in. This was obtained employing NPT dynamics as provided by ASE to generate 10000 images with variety. Of this total, 50 representatives were randomly chosen from the images after thermalization. See Figure 54 for the thermalization pattern.
2. **Compute the reference** values for the energies, forces and stresses. For this step we executed single point self-consistent calculations on each representative of the samples, using a Projector-Augmented plane Wave (PAW) Perdew-Burke-Ernzerhof (PBE) DFT pseudo-potential through Vienna Ab initio Simulation Package (VASP). The output files are then converted, joined and shuffled to compose the training and test sets for the machine learning procedure.
3. **Fine tune** the foundation model on the new dataset; in our case MACE-MP-0 was chosen.[10] The training and test sets are the input data for the training of the neural network that underlies MACE. The training spans 2000 epochs, at the end of which the compiled model is obtained, along with statistics. Plots showing neural network loss and error on energy versus epochs are available in Figure 55.

[10]Reference procedures are available at https://github.com/ACEsuit/mace?tab=readme-ov-file#finetuning-foundation-models

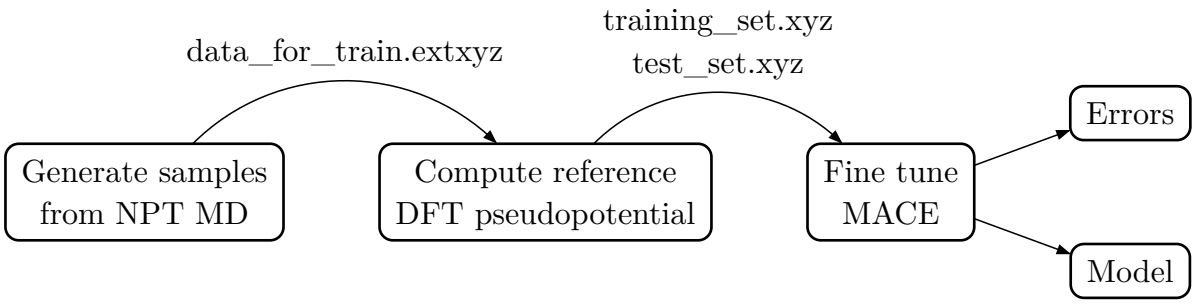


Figure 53: Flow diagram for fine-tuning of MACE.

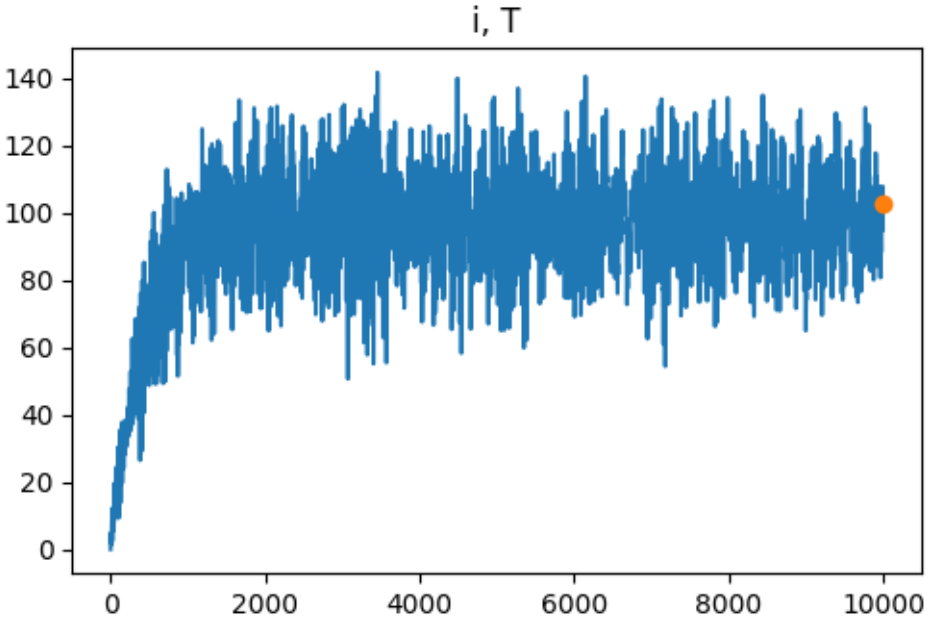


Figure 54: Temperature during sample generation.

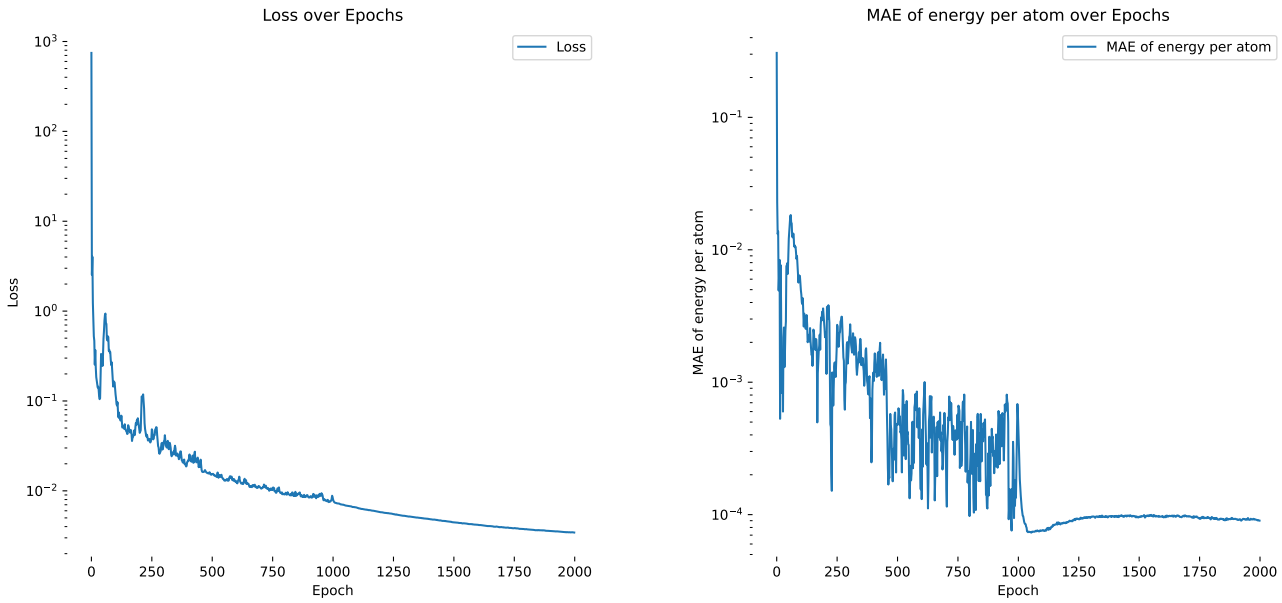


Figure 55: Loss (left) and MAE of the energy (right) over epochs plots for the fine-tuning of a new model on ice Ih, with MACE-MP-0 small as foundation model.

5.4 PHONONS CALCULATIONS

For phonons dispersion calculations, three main tools were tested:

- ASE built-in Phonons class. The current implementation of phonons calculation in ASE[11] is outclassed by Phonopy, partly because the former does not take account of symmetries.[12]
- Phonopy [19] is not native to ASE, so it needed a conversion interface for the atoms positions and forces, that was found online.[13]
- PHON [20] is not native to ASE, and it also needed a conversion interface for the atoms positions and forces; that was not found, so a conversion interface between PHON and ASE was written for our specific purposes.[14]

[11] https://wiki.fysik.dtu.dk/ase/ase/phonons.html

[12] See https://gitlab.com/ase/ase-workshop-discussion/-/issues/7#note_245747917 and https://gitlab.com/ase/ase/-/issues/1235

[13] https://gitlab.com/drFaustroll/lavello/-/blob/01296fa0b4530f07cfb97a02358bdfa289f8199f/phonons.py

[14] https://github.com/visika/tesi-magistrale/blob/32d5e30b73990daacb712e37872c22b89586f570/simulazioni/02_water/04_crystal_phonons/phon/00.template/forces.py

6 | CONCLUSIONS

In this thesis work we learned how to use and implement in our computations the MACE MLP calculator. This allowed fast and accurate prototyping of different molecules configurations, phonon dispersions and ice polymorphs lattice energies, with orders of magnitude shorter execution times with respect to DFT methods. It is noteworthy that test simulations could be executed effortlessly on a high-end laptop CPU, for a tight feedback loop between script creation and debugging. The results obtained using MACE show a satisfiying agreement with the base models onto which it is trained, with the exception of the phonon dispersions, which show a slight over-estimation of the energies, while the shape of the graphs remain accurate. Overall MLPs demonstrate a convenient accuracy/cost ratio; the combination of this charactheristic with the ability to fine-tune models to better reproduce the behaviour of particular structures of one's interest, makes this approach very versatile, and a precious addition in the toolbox of the material scientists.

7 | ACKNOWLEDGMENTS

I would like to express my gratitude to Flaviano Della Pia for his invaluable assistance in learning the new tools and for providing me with insightful advice and practical tutorials. I am also indebted to Andrea Zen for his meticulous guidance and prompt advice. Finally, I would like to thank Dario Alfè for imparting me with the fundamentals of knowledge in this field and for steering my thesis in the right direction.

This work has been funded by project code PIR01_00011 “IBISCo”, PON 2014-2020, for all three entities (INFN, UNINA and CNR).

BIBLIOGRAPHY

[1] S. Karki, T. Friščić, L. Fábián, P. R. Laity, G. M. Day, and W. Jones, "Improving Mechanical Properties of Crystalline Solids by Cocrystal Formation: New Compressible Forms of Paracetamol," *Advanced Materials*, vol. 21, no. 38–39, pp. 3905–3909, 2009, doi: 10.1002/adma.200900533.

[2] J. Behler, "Four Generations of High-Dimensional Neural Network Potentials," *Chemical Reviews*, vol. 121, no. 16, pp. 10037–10072, Aug. 2021, doi: 10.1021/acs.chemrev.0c00868.

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 1263–1272. Accessed: Jul. 13, 2024. [Online]. Available: https://proceedings.mlr.press/v70/gilmer17a.html

[4] S. Raschka, Y. Liu, V. Mirjalili, and D. Dzhulgakov, *Machine Learning with PyTorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*, 1st edition. Packt Publishing, 2022.

[5] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. Wiltschko, "A Gentle Introduction to Graph Neural Networks," *Distill*, vol. 6, no. 8, p. 10–11, Aug. 2021, doi: 10.23915/distill.00033.

[6] D. Eisenberg and W. Kauzmann, *The Structure and Properties of Water*. Oxford University Press, 2005. doi: 10.1093/acprof:oso/9780198570264.001.0001.

[7] W. Klopper, J. G. C. M. v. D.-v. de Rijdt, and F. B. van Duijneveldt, "Computational Determination of Equilibrium Geometry and Dissociation Energy of the Water Dimer," *Physical Chemistry Chemical Physics*, vol. 2, no. 10, pp. 2227–2234, Jan. 2000, doi: 10.1039/A910312K.

[8] R. Kalescky, W. Zou, E. Kraka, and D. Cremer, "Local Vibrational Modes of the Water Dimer – Comparison of Theory and Experiment," *Chemical Physics Letters*, vol. 554, pp. 243–247, Dec. 2012, doi: 10.1016/j.cplett.2012.10.047.

[9] T. R. Dyke, K. M. Mack, and J. S. Muenter, "The Structure of Water Dimer from Molecular Beam Electric Resonance Spectroscopy," *The Journal of Chemical Physics*, vol. 66, no. 2, pp. 498–510, Jan. 1977, doi: 10.1063/1.433969.

[10] A. Mukhopadhyay, S. S. Xantheas, and R. J. Saykally, "The Water Dimer II: Theoretical Investigations," *Chemical Physics Letters*, vol. 700, pp. 163–175, May 2018, doi: 10.1016/j.cplett.2018.03.057.

[11] L. A. Curtiss, D. J. Frurip, and M. Blander, "Studies of Molecular Association in H2O and D2O Vapors by Measurement of Thermal Conductivity," *The Journal of Chemical Physics*, vol. 71, no. 6, pp. 2703–2711, Sep. 1979, doi: 10.1063/1.438628.

[12] F. Della Pia, A. Zen, D. Alfè, and A. Michaelides, "DMC-ICE13 : Ambient and High Pressure Polymorphs of Ice from Diffusion Monte Carlo and Density Functional Theory," *The Journal of Chemical Physics*, vol. 157, no. 13, p. 134701–134702, Oct. 2022, doi: 10.1063/5.0102645.

[13] C. G. Salzmann, "Advances in the Experimental Exploration of Water's Phase Diagram," *The Journal of Chemical Physics*, vol. 150, no. 6, p. 60901–60902, Feb. 2019, doi: 10.1063/1.5085163.

[14] M. K. Gupta *et al.*, "Phonons and Anomalous Thermal Expansion Behavior of H2O and D2O Ice Ih," *Physical Review B*, vol. 98, no. 10, p. 104301–104302, Sep. 2018, doi: 10.1103/PhysRevB.98.104301.

[15] W. B. Holzapfel and S. Klotz, "Coherent Thermodynamic Model for Ice Ih —A Model Case for Complex Behavior," *The Journal of Chemical Physics*, vol. 155, no. 2, p. 24506–24507, Jul. 2021, doi: 10.1063/5.0049215.

[16] T. Strässle, A. M. Saitta, S. Klotz, and M. Braden, "Phonon Dispersion of Ice under Pressure," *Physical Review Letters*, vol. 93, no. 22, p. 225901–225902, Nov. 2004, doi: 10.1103/PhysRevLett.93.225901.

[17] R. N. Barnett and U. Landman, "Born-Oppenheimer Molecular-Dynamics Simulations of Finite Systems: Structure and Dynamics of (H 2 O) 2," *Physical Review B*, vol. 48, no. 4, pp. 2081–2097, Jul. 1993, doi: 10.1103/PhysRevB.48.2081.

[18] M. Sprik, J. Hutter, and M. Parrinello, "Ab\hphantom{,}Initio Molecular Dynamics Simulation of Liquid Water: Comparison of Three Gradient-corrected Density Functionals," *The Journal of Chemical Physics*, vol. 105, no. 3, pp. 1142–1152, Jul. 1996, doi: 10.1063/1.471957.

[19] A. Togo, "First-Principles Phonon Calculations with Phonopy and Phono3py," *Journal of the Physical Society of Japan*, vol. 92, no. 1, p. 12001–12002, Jan. 2023, doi: 10.7566/JPSJ.92.012001.

[20] D. Alfè, "PHON: A Program to Calculate Phonons Using the Small Displacement Method," *Computer Physics Communications*, vol. 180, no. 12, pp. 2622–2633, Dec. 2009, doi: 10.1016/j.cpc.2009.03.010.

[21] N. Blagden, M. de Matas, P. T. Gavan, and P. York, "Crystal Engineering of Active Pharmaceutical Ingredients to Improve Solubility and Dissolution Rates," *Advanced Drug Delivery Reviews*, vol. 59, no. 7, pp. 617–630, Jul. 2007, doi: 10.1016/j.addr.2007.05.011.

[22] C. M. Reddy, G. Rama Krishna, and S. Ghosh, "Mechanical Properties of Molecular Crystals—Applications to Crystal Engineering," *CrystEngComm*, vol. 12, no. 8, p. 2296–2297, 2010, doi: 10.1039/c003466e.

[23] D. M. S. Martins *et al.*, "Temperature- and Pressure-Induced Proton Transfer in the 1:1 Adduct Formed between Squaric Acid and 4,4\prime-Bipyridine", *Journal of the American Chemical Society*, vol. 131, no. 11, pp. 3884–3893, Mar. 2009, doi: 10.1021/ja8082973.

[24] A. M. Reilly and A. Tkatchenko, "Understanding the Role of Vibrations, Exact Exchange, and Many-Body van Der Waals Interactions in the Cohesive Properties of Molecular Crystals," *The Journal of Chemical Physics*, vol. 139, no. 2, p. 24705–24706, Jul. 2013, doi: 10.1063/1.4812819.

[25] S. L. Price, "The Computational Prediction of Pharmaceutical Crystal Structures and Polymorphism," *Advanced Drug Delivery Reviews*, vol. 56, no. 3, pp. 301–319, Feb. 2004, doi: 10.1016/j.addr.2003.10.006.

[26] I. Batatia *et al.*, "A Foundation Model for Atomistic Materials Chemistry." Accessed: Jan. 11, 2024. [Online]. Available: http://arxiv.org/abs/2401.00096

[27] C. Schran, F. L. Thiemann, P. Rowe, E. A. Müller, O. Marsalek, and A. Michaelides, "Machine Learning Potentials for Complex Aqueous Systems Made Simple," *Proceedings of the National Academy of Sciences*, vol. 118, no. 38, p. e2110077118, Sep. 2021, doi: 10.1073/pnas.2110077118.

[28] O. Björneholm *et al.*, "Water at Interfaces," *Chemical Reviews*, vol. 116, no. 13, pp. 7698–7726, Jul. 2016, doi: 10.1021/acs.chemrev.6b00045.

[29] V. Kapil and E. A. Engel, "A Complete Description of Thermodynamic Stabilities of Molecular Crystals," *Proceedings of the National Academy of Sciences*, vol. 119, no. 6, p. e2111769119, Feb. 2022, doi: 10.1073/pnas.2111769119.

[30] B. Cheng, E. A. Engel, J. Behler, C. Dellago, and M. Ceriotti, "Ab Initio Thermodynamics of Liquid and Solid Water," *Proceedings of the Na-*

tional Academy of Sciences, vol. 116, no. 4, pp. 1110–1115, Jan. 2019, doi: 10.1073/pnas.1815117116.

[31] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," *Physical Review*, vol. 136, no. 3B, p. B864–B871, Nov. 1964, doi: 10.1103/PhysRev.136.B864.

[32] W. Kohn and L. J. Sham, "Self-Consistent Equations Including Exchange and Correlation Effects," *Physical Review*, vol. 140, no. 4A, p. A1133–A1138, Nov. 1965, doi: 10.1103/PhysRev.140.A1133.

[33] D. Alfè, "Notes on Statistical and Computational Physics," 2023.

[34] L. Verlet, "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Physical Review*, vol. 159, no. 1, pp. 98–103, Jul. 1967, doi: 10.1103/PhysRev.159.98.

[35] H. C. Andersen, "Molecular Dynamics Simulations at Constant Pressure and/or Temperature," *The Journal of Chemical Physics*, vol. 72, no. 4, pp. 2384–2393, Feb. 1980, doi: 10.1063/1.439486.

[36] Daan Frenkel and Berend Smit, *Understanding Molecular Simulation*. Elsevier, 2002. doi: 10.1016/B978-0-12-267351-1.X5000-7.

[37] G. J. O. Beran, J. D. Hartman, and Y. N. Heit, "Predicting Molecular Crystal Properties from First Principles: Finite-Temperature Thermochemistry to NMR Crystallography," *Accounts of Chemical Research*, vol. 49, no. 11, pp. 2501–2508, Nov. 2016, doi: 10.1021/acs.accounts.6b00404.

[38] A. Otero-de-la-Roza and E. R. Johnson, "A Benchmark for Non-Covalent Interactions in Solids," *The Journal of Chemical Physics*, vol. 137, no. 5, p. 54103–54104, Aug. 2012, doi: 10.1063/1.4738961.

[39] A. Zen, J. G. Brandenburg, J. Kliměs, A. Tkatchenko, D. Alfè, and A. Michaelides, "Fast and Accurate Quantum Monte Carlo for Molecular Crystals," *Proceedings of the National Academy of Sciences*, vol. 115, no. 8, pp. 1724–1729, Feb. 2018, doi: 10.1073/pnas.1715434115.

[40] E. Whalley, "Energies of the Phases of Ice at Zero Temperature and Pressure," *The Journal of Chemical Physics*, vol. 81, no. 9, pp. 4087–4092, Nov. 1984, doi: 10.1063/1.448153.

[41] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, "A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu," *The Journal of Chemical Physics*, vol. 132, no. 15, p. 154104–154105, Apr. 2010, doi: 10.1063/1.3382344.

[42] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to Simulate Complex Physics with Graph Networks," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 8459–8468. Accessed: Jul. 13, 2024. [Online]. Available: https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html

[43] P. W. Battaglia *et al.*, "Relational Inductive Biases, Deep Learning, and Graph Networks." Accessed: Jul. 13, 2024. [Online]. Available: http://arxiv.org/abs/1806.01261

[44] "Physical Chemistry of Water." Accessed: May 18, 2024. [Online]. Available: https://web.archive.org/web/20201020055601/https://msu.edu/course/css/850/snapshot.afs/teppen/physical_chemistry_of_water.htm

[45] A. Mukhopadhyay, W. T. S. Cole, and R. J. Saykally, "The Water Dimer I: Experimental Characterization," *Chemical Physics Letters*, vol. 633, pp. 13–26, Jul. 2015, doi: 10.1016/j.cplett.2015.04.016.

[46] M. Schütz, S. Brdarski, P.-O. Widmark, R. Lindh, and G. Karlström, "The Water Dimer Interaction Energy: Convergence to the Basis Set Limit at the Correlated Level," *The Journal of Chemical Physics*, vol. 107, no. 12, pp. 4597–4605, Sep. 1997, doi: 10.1063/1.474820.

[47] P. Hobza, O. Bludský, and S. Suhai, "Reliable Theoretical Treatment of Molecular Clusters: Counterpoise-corrected Potential Energy Surface and Anharmonic Vibrational Frequencies of the Water Dimer," *Physical Chemistry Chemical Physics*, vol. 1, no. 13, pp. 3073–3078, Jan. 1999, doi: 10.1039/A902109D.

[48] P. Flubacher, A. J. Leadbetter, and J. A. Morrison, "Heat Capacity of Ice at Low Temperatures," *The Journal of Chemical Physics*, vol. 33, no. 6, pp. 1751–1755, Dec. 1960, doi: 10.1063/1.1731497.

[49] A. J. Leadbetter, "The Thermodynamic and Vibrational Properties of H2O Ice and D2O Ice," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 287, no. 1410, pp. 403–425, Sep. 1965, doi: 10.1098/rspa.1965.0187.

[50] L. B. Skinner, C. Huang, D. Schlesinger, L. G. M. Pettersson, A. Nilsson, and C. J. Benmore, "Benchmark Oxygen-Oxygen Pair-Distribution Function of Ambient Water from x-Ray Diffraction Measurements with a Wide Q-range," *The Journal of Chemical Physics*, vol. 138, no. 7, p. 74506–74507, Feb. 2013, doi: 10.1063/1.4790861.

[51] "Wiki:Archit_ib_en [Ibisco HPC Wiki]." Accessed: Jul. 12, 2024. [Online]. Available: https://ibiscohpc-wiki.scope.unina.it/wiki:archit_ib_en

[52] I. Batatia, D. P. Kovacs, G. N. C. Simm, C. Ortner, and G. Csanyi, "MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=YPpSngE-ZU

[53] I. Batatia *et al.*, "The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials," no. arXiv:2205.06643. 2022. doi: 10.48550/arXiv.2205.06643.

[54] I. Batatia *et al.*, "The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials." Accessed: Jun. 02, 2024. [Online]. Available: https://arxiv.org/abs/2205.06643

[55] J. P. Darby *et al.*, "Tensor-Reduced Atomic Density Representations," *Physical Review Letters*, vol. 131, no. 2, p. 28001–28002, Jul. 2023, doi: 10.1103/PhysRevLett.131.028001.

[56] I. Batatia, D. P. Kovács, G. N. C. Simm, C. Ortner, and G. Csányi, "MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields," 2022.

[57] G. Dusson *et al.*, "Atomic Cluster Expansion: Completeness, Efficiency and Stability," *Journal of Computational Physics*, vol. 454, p. 110946–110947, Apr. 2022, doi: 10.1016/j.jcp.2022.110946.

[58] D. P. Kovács, I. Batatia, E. S. Arany, and G. Csányi, "Evaluation of the MACE Force Field Architecture: From Medicinal Chemistry to Materials Science," *The Journal of Chemical Physics*, vol. 159, no. 4, p. 44118–44119, Jul. 2023, doi: 10.1063/5.0155322.

[59] H. Kaur *et al.*, "Data-Efficient Fine-Tuning of Foundational Models for First-Principles Quality Sublimation Enthalpies." Accessed: Jun. 11, 2024. [Online]. Available: http://arxiv.org/abs/2405.20217