

École polytechnique de Louvain

Accelerating ab initio relaxations using machine-learning interatomic potentials

Author : **Anton DENISENKO**

Supervisors : **Gian-Marco RIGNANESE, Matteo GIANTOMASSI**

Readers : **Benoît HACKENS, Flavio ABREU ARAUJO, Xavier GONZE**

Academic year 2025–2026

Master [120] in Chemical and Materials Engineering

Abstract

Ab initio materials simulations, used for a wide class of calculations of technological interest, rely on computationally costly DFT methods, making them unaffordable for systems of larger sizes. Machine-learning interatomic potentials (MLIPs) have been developed as an alternative solution, attempting to approach the accuracy of DFT, while keeping the computational cost similar to empirical potentials.

The objective of this work is to implement universal machine-learning interatomic potentials within the computational framework of ABINIT. To this purpose, we evaluate three related MLIPs based on the Point Edge Transformer architecture in terms of accuracy of their predictions and improvements in computational speed. In the later stages, we discuss implementation strategies for a module allowing to use these potentials for calculations directly in ABINIT.

Contents

Glossary	4
List of Tables	5
List of Figures	6
I Introduction	7
II Background	8
II.1 Density Functional Theory (DFT)	8
II.2 Structural relaxation	11
II.3 Machine learning interatomic potentials (MLIPs)	12
II.4 Computational packages	15
III Feasibility studies	17
III.1 Correctness of UPET calculations	17
III.2 Efficiency of UPET in structure relaxation	23
IV Implementation considerations	30
V Conclusion	33
Acknowledgements	33
Bibliography	34
Appendix A: Relevant data for feasibility studies	39
Appendix B: CalcBuilder implementation for PET-OAM-XL	41
Appendix C: ABINIT scripts used in calculations	42

Glossary

API – Application Programming Interface
ASE – Atomic Simulation Environment
BFGS – Broyden–Fletcher–Goldfarb–Shanno (algorithm)
CPU – Central Processing Unit
DFT – Density Functional Theory
EOS – Equation of State
GGA – Generalised Gradient Approximation
GNN – Graph Neural Network
GPU – Graphics Processing Unit
IQR – Inter-Quantile Range
LDA/LSDA – Local (Spin) Density Approximation
MAE – Mean Absolute Error
MPNN – Message-Passing Neural Network
PBE – Perdew–Burke–Ernzerhof (exchange-correlation potential)
PES – Potential Energy Surface
RMSD – Root Mean Square Deviation
SCF – Self-Consistent Field
uMLIP – (Universal) Machine-Learning Interatomic Potential
UPET – Unconstrained Point Edge Transformer

List of Tables

1	Characteristics of the computational devices used in the study	23
2	Average time required for a step of the BFGS optimisation algorithm in the 16-atom diamond Si experiment, on PC and Lemaitre4	24
3	Effect of PET-OAM-L on the relaxation of the perturbed 54-atom Si supercell in ABINIT	27
4	Effect of PET-OAM-L on the relaxation of the perturbed conventional cell of P ₂ O ₅ in ABINIT	28

List of Figures

1	Point Edge Transformer (PET) architecture. White and coloured boxes represent layers, grey ones represent data	14
2	Boxplot showing the relative error in calculations of EOS parameters between the selected UPET family potentials and the reference all-electron dataset	18
3	Average value of ν for the converged structures of the chemical elements in the dataset, for PET-OAM-L. Plot created with pymatviz [1]	19
4	Number of structures with failures of convergence of the EOS, for PET-OAM-L. Plot created with pymatviz [1]	20
5	Representations of different values of ν in the calculations of EOS parameters with PET-OAM-L	21
6	Boxplot showing the relative error in calculations of EOS parameters between the older GNN-based potentials and the reference all-electron results, from the study [2]. Reused under CC BY 4.0	22
7	Parity plots of forces, energies and stresses of 15 perturbed diamond silicon configurations, between ABINIT-PBE and PET-OAM-L calculations	23
8	Time required for a BFGS step in ABINIT for a perturbed 16-atom diamond Si supercell. Calculations performed on a) a personal computer and b) cluster Lemaitre4	24
9	Effect of PET-OAM-L on the number of BFGS relaxation steps in ABINIT in the 16-atom experiment on Si	25
10	Time required for a BFGS step in ABINIT for a perturbed 54-atom diamond Si supercell, with and without prior relaxation with OAM-L	26
11	Three-dimensional view of three diamond Si supercells: the unperturbed configuration, one optimised only with ABINIT and one optimised with OAM and ABINIT	27
12	Time required for a BFGS step in ABINIT for a perturbed P_2O_5 cell, with and without prior relaxation with OAM-L	28
13	Average value of ν and failures of convergence of EOS by chemical element for PET-MAD-S.	39
14	Average value of ν and failures of convergence of EOS by chemical element for PET-OMAD-L.	40

I Introduction

Computational approaches have become essential to materials science. Materials simulations help facilitate material design and discovery, allow researchers to study reaction mechanisms and determine physical properties, without resorting to experimental testing, often expensive or impractical for a given task. Ab initio materials simulations rely on the Born-Oppenheimer approximation, where the potential governing the nuclei is represented by electronic energy dependent on the atomic configuration, known as the potential energy surface.

Knowledge of the potential energy surface allows the identification of stable atomic configurations and is necessary to compute materials properties [3]. Consequently, there is a strong need for the PESs to be accurately generated, which is possible via density-functional theory (DFT) methods [4]. However, they rely on quantum mechanical calculations, whose computational cost scales in a cubic manner with the system size, which is unreasonable in many cases of scientific interest, such as surfaces or amorphous solids.

Empirical interatomic potentials, which establish a parametrised dependency between physical properties and atomic configurations, have been widely used for large-scale studies. [5]. Such potentials are unable to reach DFT accuracy in systems with complex electron interactions, and they usually have a limited transferability, with approximations tailored to a specific application. As such, they cannot be considered a proper replacement for DFT methods.

Machine-learning interatomic potentials (MLIPs) were developed as an intermediate solution, with a computational cost comparable to that of analytical potentials but with much higher accuracy, striving to compete with DFT methods [6]. The gain in accuracy is explained by a direct fit to datasets generated with DFT, which originally required material specialisation, but with the advent of universal MLIPs based on neural networks that is no longer required [2].

ABINIT [7] is a popular computational package for atomistic simulations. Among others, it implements DFT methods for energy and force calculation. While it is possible to use MLIPs with ABINIT indirectly, via third-party simulation packages such as ASE [8] or by directly passing atomic configurations from one framework to another, we are interested in making it possible for such calculations to be done directly in ABINIT.

The objective of this work is to implement universal machine-learning interatomic potentials within the computational framework of ABINIT. To this purpose, we assess three high-performing uMLIPs using the Point Edge Transformer architecture [9]: we examine the correctness of their outputs, and the speed-up that can be achieved by combining them with DFT methods. We will discuss the codebase and implementation strategies, with the goal to eventually make a module allowing to use them directly within ABINIT.

II Background

II.1 Density Functional Theory (DFT)

Density Functional Theory is a computational quantum mechanical modelling method used to study the electronic structure of complex many-body systems. The predictions made by this method allow to study materials and test scientific theories from first principles – electron states and their interactions with nuclei. Modelling helps us understand mechanisms of physical phenomena, discover new structures or analyse their behaviour without resorting to experimental data. It has achieved great success in the recent decades, with the number of publications involving DFT rising from 1100 to 5600 between 1996 and 2006 [4].

Within the Born-Oppenheimer approximation, nuclei and electrons are treated as separate mathematical problems. With M nuclei at coordinates $\mathbf{r}_1, \dots, \mathbf{r}_M$, we can express the ground-state energy as $E(\mathbf{r}_1, \dots, \mathbf{r}_M)$, a function known as the adiabatic potential energy surface (PES) of the atoms. The energy of a system with multiple electrons and nuclei is described by the time-independent Schrödinger equation:

$$\left[\frac{h^2}{2m} \sum_{i=1}^N \nabla_i^2 + \sum_{i=1}^N V(\mathbf{r}_i) + \sum_{i=1}^N \sum_{j<i} U(\mathbf{r}_i, \mathbf{r}_j) \right] \Psi = E\Psi, \quad (1)$$

where m is the electron mass, V is the interaction energy between an electron and the collection of nuclei, U is the electron-electron interaction energy and N is the number of electrons in the system. Ψ is the complex-valued electronic wave function of the spatial coordinates of the system's electrons: $\Psi(\mathbf{r}_1, \dots, \mathbf{r}_N)$. It can be approximated as a product of individual wave functions $\Psi = \psi(\mathbf{r}_1)\psi(\mathbf{r}_2)\dots\psi(\mathbf{r}_N)$, however even in this form solving this equation is extraordinarily difficult, as the dimensionality of the system makes it unapproachable: for a CO_2 molecule with 22 electrons, the wave function is a 66-dimensional function, for a system of 100 atoms the number of dimensions would be in the thousands.

Solving the Schrödinger equation is a fundamental problem in quantum physics, however from a practical standpoint the wave function cannot be observed for any set of coordinates. The quantity that can be measured is electron density:

$$n(\mathbf{r}) = 2 \sum_i \Psi_i^*(\mathbf{r}) \Psi_i(\mathbf{r}) \quad (2)$$

Notably, this is a three-dimensional function that still contains a great amount of energy about the system. According to the first Hohenberg-Kohn theorem: “the ground-state energy from the Schrödinger equation is a unique functional of the electron density”. Ground-state energy density determines the energy and wave function of the ground state, meaning we can express it as $E[n(\mathbf{r})]$ and focus on finding the functional of electron density. Second Hohenberg-Kohn theorem describes one of its properties: “the electron density that minimises

the energy of the overall functional is the true electron density corresponding to the full solution of the Schrödinger equation". The energy for the true functional form could be minimised by varying the electron density, which would then tell us the ground state.

As single-electron wave functions collectively define electron density, they present a way to express the energy functional:

$$E[\{\psi_i\}] = E_{\text{known}}[\{\psi_i\}] + E_{\text{XC}}[\{\psi_i\}] \quad (3)$$

E_{known} includes electron kinetic energies and Coulomb interactions (electron-electron, nuclei-nuclei and electron-nuclei). E_{XC} is referred to as the exchange-correlation functional, which includes all quantum mechanical effects unaccounted for in E_{known} .

The difficulty with finding the right electron density was solved by Kohn and Sham, as the task was expressed in a set of equations only involving single-electron wave functions. The Kohn-Sham equations have the following form [10]:

$$\left[-\frac{\hbar^2}{2m} \nabla^2 + V(\mathbf{r}) + V_H(\mathbf{r}) + V_{\text{XC}}(\mathbf{r}) \right] \psi_i(\mathbf{r}) = \varepsilon_i \psi_i(\mathbf{r}), \quad (4)$$

where V describes the interaction between an electron and the set of nuclei, V_H is the Hartree potential describing the Coulomb repulsion between the currently considered electron and the total electron density, and V_{XC} is the functional derivative of the exchange-correlation energy with respect to electron density.

As the Hartree potential requires knowledge of the electron density, which we would only get by solving the Kohn-Sham equations, solving this set of equations is best represented as an iterative problem. They are solved via a self-consistent field (SCF) cycle:

1. As the initial guess, a trial electron density $n(\mathbf{r})$ is defined.
2. The Kohn-Sham equations are solved using this trial density to find the single-electron wave functions $\psi_i(\mathbf{r})$, and the electron density $n_{\text{KS}}(\mathbf{r})$ is calculated using equation 2.
3. The calculated electron density $n_{\text{KS}}(\mathbf{r})$ is compared with the one used for calculations ($n(\mathbf{r})$). If the two are the same, then the ground-state density has been found and can be used to compute the total energy. Otherwise, the trial electron density is updated and the cycle continues.

Perdew–Burke–Ernzerhof (PBE) exchange-correlation functional

The simplest approximation of the unknown true Kohn-Sham functional is the local density approximation (LDA), which defines the local exchange-correlation potential as the exchange potential of spatially uniform electron gas, at the electron density equivalent to that

seen in the system. It accounts for the uniform density limit of the Kohn-Sham functional: at uniform density the exchange-correlation energy must be the energy of electron gas of that density. The local density approximation is successful in bulk solids, but not in predictions of properties of atoms and molecules. Local spin density approximation (LSDA) is a derivative of LDA that permits different spin-up and spin-down electron densities.

The next step is the generalised gradient approximation (GGA). As real electron densities are not uniform, it is necessary to account for the spatial variation in electron density to create a more performant functional. So, the exchange-correlation functional in GGA depends on both the local electron density and its gradient, which lacks a single form unlike LDA [11]:

$$V_{\text{XC}}^{\text{GGA}}(\mathbf{r}) = V_{\text{XC}}[n(\mathbf{r}), \nabla n(\mathbf{r})]. \quad (5)$$

The Perdew–Burke–Ernzerhof (PBE) [12] exchange-correlation functional is a widely-used non-empirical functional in the GGA group [13]. It was developed with the goal to improve the parametrisation of the numerical GGA correlation energy functional, while reproducing LSDA response of the density of uniform electron gas and other energetically significant properties.

GGA energy can be expressed in the following form:

$$E_{\text{XC}}^{\text{GGA}}[n_{\uparrow}, n_{\downarrow}] = \int d^3r n(\mathbf{r}) \epsilon_{\text{X}}^{\text{unif}}(n) F_{\text{XC}}(r_s, \zeta, s), \quad (6)$$

where $\epsilon_{\text{X}}^{\text{unif}}(n)$ is the per-electron exchange energy of uniform electron gas with density n , $F_{\text{XC}}(r_s, \zeta, s) = F_{\text{X}} + F_{\text{C}}$ is the exchange-correlation enhancement factor that accounts for spatial variation in density, r_s is the Wigner-Seitz radius, ζ is the spin polarisation and s is the reduced density gradient.

To account for the LSDA behaviour at the uniform density limit and for the Lieb-Oxford inequality, the exchange factor in the PBE functional is defined as:

$$F_{\text{X}}(s) = 1 + \kappa - \frac{\kappa}{1 + \mu s^2 / \kappa}, \quad (7)$$

where $\kappa = 0.804$ and $\mu \approx 0.2195$.

Pseudopotentials

The most important electrons for materials simulations are the valence electrons, while those on lower orbitals are comparatively unimportant, however they still generate a large number of basis functions for a proper description of electron-electron repulsion. In the lower half of the periodic table this problem strongly affects computation efficiency, with relativistic effects further complicating matters.

These problems can be mitigated by modelling the core electrons as a single function, keeping only the valence electrons explicit in the calculations. This function is referred to as an Effective Core Potential, or Pseudopotential. This simplification allows to get good results at a much lower computational cost. Norm-conserving pseudopotentials maintain that the square of the reference wave function and that of the pseudo-wave from 0 to the core radius r_c are the same. Those functions are obtained from the chosen functional, like LSDA or PBE [14].

II.2 Structural relaxation

The potential energy surface (PES) describes the energy of the system. As energy is a function of atomic positions ($E(\mathbf{R}) = E(\mathbf{r}_1, \dots, \mathbf{r}_N)$), which we represent as Cartesian coordinates, the dimensionality of said function is $3N$ for N atoms in the system. Structural relaxation is the optimisation of this multi-dimensional function: should we manage to find its global minimum, we also find the equilibrium configuration of the system [14].

In order to do so, we need to know at least the gradient $\mathbf{g}$ of the system. All computational methods assume that the gradient can be computed analytically, and there are algorithms which only utilise the gradient to perform optimisation, such as the Steepest Descent and the Conjugate Gradient methods, however, as they do not have access to information of the second derivative, they are only able to locate minima (not saddle points) and thus converge linearly, slowing down further when near a minimum.

In order to improve convergence, we also need the Hessian $\mathbf{H}$ of the energy function. The Newton-Raphson (NR) method utilises it as part of the second-order Taylor decomposition of the function:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \mathbf{g}^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{H}(\mathbf{x} - \mathbf{x}_0). \quad (8)$$

At a stationary point, $f(\mathbf{x}) = 0$, thus we get the NR step:

$$(\mathbf{x} - \mathbf{x}_0) = -\mathbf{H}^{-1}\mathbf{g}. \quad (9)$$

In the coordinate system with the basis composed of the Hessian eigenvectors, the NR step takes the form $\Delta x'_i = -\frac{f_i}{\varepsilon_i}$, where f_i is the projection of the gradient on the Hessian eigenvector, with eigenvalue ε_i . The convergence of this algorithm is quadratic near a stationary point (specifically, in the region where the Taylor decomposition is valid), however computing the Hessian itself, especially in systems with a large number of atoms, can constitute a problem. Instead of direct computation, we can use the updating scheme using an approximate Hessian, in particular the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [15]:

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \Delta\mathbf{H} \quad (10)$$

$$\Delta\mathbf{H}_{\text{BFGS}} = \frac{\Delta\mathbf{g} \Delta\mathbf{g}^T}{\Delta\mathbf{g}^T \Delta\mathbf{x}} - \frac{\mathbf{H} \Delta\mathbf{x} \Delta\mathbf{x}^T \mathbf{H}}{\Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x}}. \quad (11)$$

Iterative updates of the Hessian require an initial guess, which can be as simple as an identity matrix (in which case the first step would be equivalent to the Steepest Descent method), calculated from low-level theory or even the exact one, which would speed up the process. In any case, the use of approximate Hessian makes the convergence of the algorithm (now referred to as quasi-NR) linear, however because the cost of calculating the Hessian is much lower, they tend to approach the stationary point more rapidly than the NR method [14].

Numerically, the exact minimum is never reached, due to floating-point errors, finite precision of the SCF procedure and the Hessian. Instead, the optimisation usually terminates when the maximum force acting on an atom falls below a set tolerance.

II.3 Machine learning interatomic potentials (MLIPs)

As electronic structure calculations are very computationally expensive, they are in practice restricted to a small number of atoms (up to a thousand) and short simulation times, significantly lower than 1 ns. For large-scale simulations, historically they have been conducted with analytic potentials connecting the structure and its energy via direct relations; they represent physical approximations but are a lot easier to compute. MLIPs were created with the goal of bridging the two: they aim for the accuracy of ab initio methods with the efficiency of functional forms.

In general, machine learning interatomic potentials employ a ML method to find a direct functional relation between the structure and its energy, they do not involve any physical approximations other than the reference electronic structure method used in its development, and they are trained on a consistent electronic structure dataset. These principles aim to guarantee rapid evaluation, generality and transferability [16].

Interatomic potentials V are also characterised by a number of invariants:

$$E = V \left((\mathbf{r}_i, a_i)_{i=1}^N \right), \quad (12)$$

where $\mathbf{r}_i$ are the atomic positions, a_i are their corresponding chemical elements and N is the number of atoms in the structure.

- **E(3)-invariance.** All finite combinations of translations, rotations and reflections in three-dimensional space. For a transformation $g \in E(3)$ acting on all atomic coordinates, $V \left((g \cdot \mathbf{r}_i, a_i)_{i=1}^N \right) = V \left((\mathbf{r}_i, a_i)_{i=1}^N \right)$

- **Permutational invariance.** Interatomic potentials are invariant with respect to atomic indices. $V \left(\dots(\mathbf{r}_i, a_i) \dots (\mathbf{r}_j, a_j) \dots \right) = V \left(\dots(\mathbf{r}_j, a_j) \dots (\mathbf{r}_i, a_i) \dots \right)$
- **Locality.** Atoms distant from each other do not interact, with few exceptions [17]. I.e. if $|\mathbf{r}_j - \mathbf{r}_i|$ is large, $V \left((\mathbf{r}_i, a_i)_{i=1}^N \right) - V \left((\mathbf{r}_i, a_i)_{i=1, i \neq n}^N \right) - V \left((\mathbf{r}_i, a_i)_{i=1, i \neq m}^N \right) + V \left((\mathbf{r}_i, a_i)_{i=1, i \neq n, m}^N \right) \approx 0$. [9]

Recently, Graph Neural Networks (GNNs), in particular message passing and equivariant architectures [2] have proved to be effective MLIPs, as they are permutationally invariant by design [18], and are able to express translational invariance and locality well. They can achieve high accuracy and be useful for generalisation, hence they became the driving force for development of universal MLIPs.

The Graph Neural Network model

A graph $\mathbf{G}$ is a pair $(\mathbf{N}, \mathbf{E})$, where $\mathbf{N}$ is the set of nodes and $\mathbf{E}$ is the set of edges. The set $ne[i]$ comprises all nodes connected to i by an edge, the set $co[i]$ – all edges that have i as a node. The vector label attached to node i is denoted as $\mathbf{l}_i$, while that attached to the edge (i_1, i_2) as $\mathbf{l}_{(i_1, i_2)}$.

The original idea behind the model was for the nodes in the graph to represent concepts, and for the edges to represent relationships. To each concept we can attach a state $\mathbf{x}_{ne[i]}$, based on the neighbourhood of i , which can then be used to produce an output $\mathbf{o}_i$, decision about the concept. Let f_ω be a parametric function (local transition function), expressing how node i depends on its neighbours and let g_ω be a parametric function called the local output function, it describes how the output is produced:

$$\mathbf{x}_i = f_\omega \left(\mathbf{l}_i, \mathbf{l}_{co[i]}, \mathbf{x}_{ne[i]}, \mathbf{l}_{ne[i]} \right), \quad (13)$$

$$\mathbf{o}_i = g_\omega(\mathbf{x}_i, \mathbf{l}_i). \quad (14)$$

We can enforce locality in the atomistic context by only constructing graph edges if the atoms are within the cut-off radius from one another. Within this framework, learning node states was done in an iterative process, converging to a fixed point [19]. This made the network computationally demanding and difficult to train, which led to the development of Message Passing NNs, where strict convergence was replaced with a finite number of message passing steps, set as a hyperparameter: based on the "senders" of messages (neighbours), their "receiver" (the atom itself) and edge labels, the hidden state is updated a set T times before computing the new labels. This allowed for easier model training, larger flexibility and variety of applications, as well as allowed more variety in message passing functions. [20]

Modern GNNs used in atomistic modelling are variants of the MPNN architecture. Due to their high accuracy and generalisation, they became prevalent in practical applications and scientific benchmarks, such as Matbench Discovery [21].

Unconstrained Point Edge Transformer models (UPET)

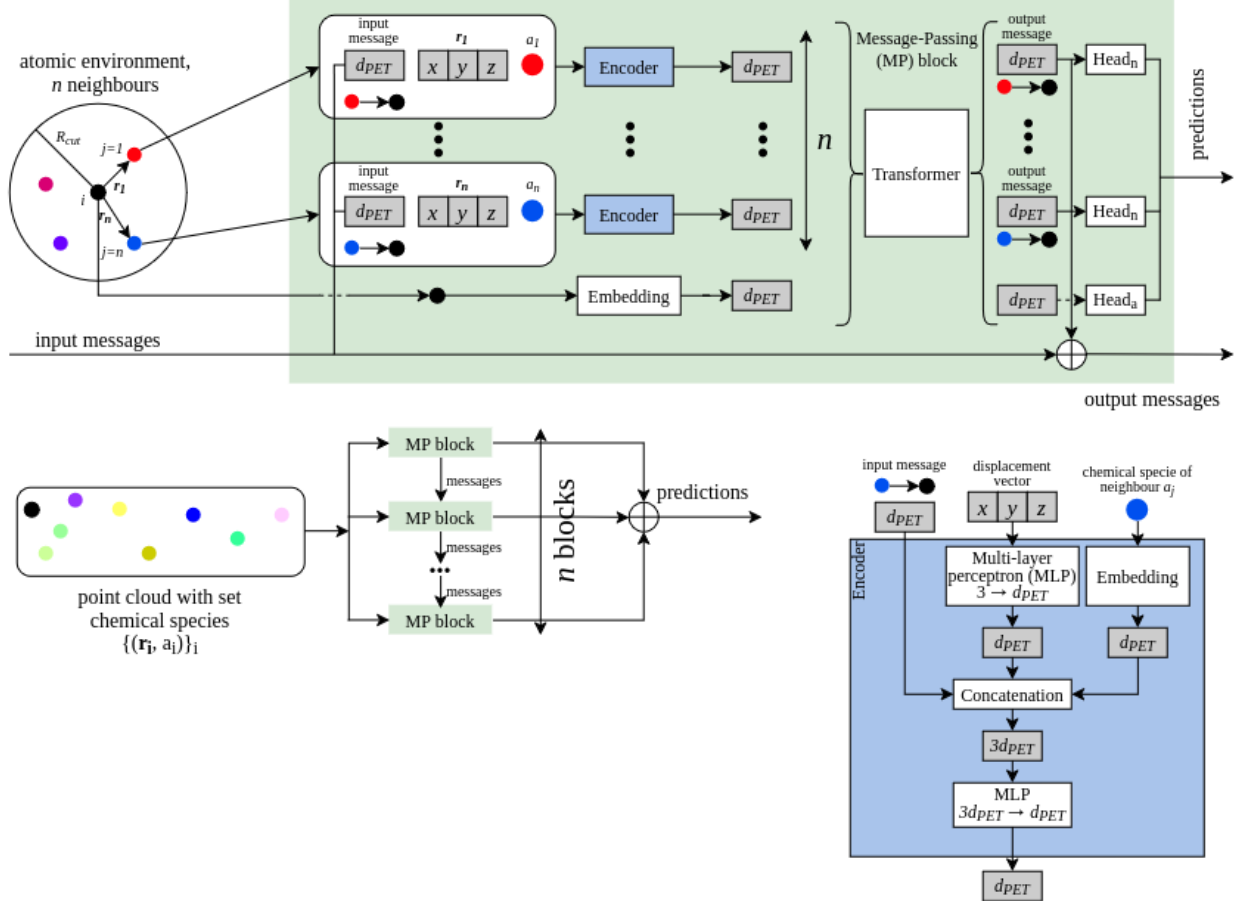


Figure 1: Point Edge Transformer (PET) architecture.

White and coloured boxes represent layers, grey ones represent data

UPET models are variants of MPNN. For each of the message passing interactions, messages are sent between all pairs of atoms within the cutoff radius of one another. Given the local atomic environment (geometry and chemical species) and the input messages from the neighbours to the central atom, the message-passing block computes property predictions and output messages to all the neighbours [22].

At first, all information associated with the neighbours is encoded to an abstract token of dimensionality d_{PET} . The collection of these tokens is transferred to the transformer, which comprises n self-attention layers, which weigh the importance of the tokens (and, thus, neighbours) in relation to others. The output tokens are treated as messages themselves [22].

The specific architecture used in the UPET machine-learned potentials is heavily based

on PET, with several key improvements:

- Modern transformer architecture is used, which involves layer normalisation and a more recent SwiGLU activation function [23].
- The number of node features for a given edge feature is increased by four: there are 30-50 times more edges than nodes in a graph representing chemical structure, so this greatly increases the number of parameters without much computational overhead.
- Models utilise an adaptive cutoff strategy: diverse datasets include materials with vastly different densities and thus the number of edges within a pre-set cutoff radius would also strongly vary. This approach ensures atomic neighbourhoods have a near-constant amount of neighbours, so that training and computational costs are roughly uniform across different structures [9].

This architecture is then applied to different materials databases. PET-OAM was trained on three datasets in different stages: MPtrj [24], Alexandria [25] and OMat [26], and at size L contains 255M parameters. PET-OMAD was trained on OMat and fine-tuned on the smaller MAD dataset [27] and also contains 255M parameters. PET-MAD was only trained on the MAD dataset and contains 25.9M parameters, thus being computationally simpler.

II.4 Computational packages

ABINIT

ABINIT [7] is an open-source computational package for materials and nanostructures that implements density functional theory and many-body perturbation theory calculations. It is able to find total energy, forces and stresses, electronic structure spectra and a wide range of physical properties from first principles. It can perform molecular dynamics, structural relaxation and a large number of other procedures relevant in materials science. It provides a number of supporting tools, such as AbiPy, a Python library for the analysis of ABINIT results.

In this study, ABINIT is used as the reference DFT calculator. For each atomic configuration, ABINIT solves the Kohn-Sham equations via SCF, then calculates energy, forces and stresses later used by the BFGS algorithm for structural optimisation. The energy calculations were performed using a norm-conserving pseudopotential based on the PBE exchange-correlation functional.

The SCF algorithm for energy calculation terminates upon reaching the convergence criterion. It can be potential/density-based, reaching the tolerance of the potential residual `tolvrs` directly connected to the SCF cycle described in II.1, or energy-based, where the

difference in energy between the current and the previous SCF iterations $|E^{(n)} - E^{(n-1)}|$ is compared against the tolerance `toldfe`. The BFGS algorithm for atomic relaxation terminates if the maximum force component $|F_{i\alpha}|, i = 1 \dots N, \alpha = \{x, y, z\}$ is below the tolerance `toldmxf`.

Atomic Simulation Environment (ASE)

The Atomic Simulation Environment [8] is a Python package for running atomistic simulations. It is intended as the front-end for atomistic simulations, having many methods for control and manipulation of the simulation environment, as well as tools for data analysis. It provides support to many DFT packages and interatomic potential codes via Calculators, which take the atomic configuration and return energies, forces and stresses.

ASE was used in this study as the interface for the UPET package. UPET provides a calculator for each of its machine-learned interatomic potentials via its own Python package, fully compatible and intended for use with ASE. The BFGS algorithm was once again used for structural relaxation, with the aforementioned force-based convergence criterion.

III Feasibility studies

In order to select the initial set of MLIPs, we turned to online benchmarks. Matbench Discovery [21] is an online leaderboard ranking MLIPs with respect to their performance in computational tasks in materials science, such as crystal stability prediction, geometry optimisation, thermal conductivity and others. The most relevant metric presented on the website for this project is labelled as RMSD, Root Mean Square Deviation between the MLIP-predicted and DFT-relaxed structures after matching.

At the time this study began (February 2026), the leading and most novel potential according to this metric was PET-OAM-XL, a potential of the UPET (Unconstrained Point Edge Transformer) family, featuring a number of models based on Point Edge Transformer architecture. As these models share their interface, we wanted to look into three subtypes: MAD-S, recommended for molecular dynamics calculations, OAM-L, a faster version of the potential on Matbench Discovery that is recommended for materials discovery tasks (including geometry optimisation), and OMAD-L, model which showed superior accuracy and transferability compared to OAM in certain use cases [9].

III.1 Correctness of UPET calculations

As the first step, we assessed these potentials using the protocol based on the Birch-Murnaghan isothermal equation of state (EOS) [28]:

$$P(V) = \frac{3B_0}{2} \left[\left(\frac{V_0}{V} \right)^{7/3} - \left(\frac{V_0}{V} \right)^{5/3} \right] \left\{ 1 + \frac{3}{4} (B_1 - 4) \left[\left(\frac{V_0}{V} \right)^{2/3} - 1 \right] \right\}, \quad (15)$$

where P is the pressure, V_0 is the equilibrium volume, V is the deformed volume, B_0 is the bulk modulus and B_1 is the derivative of the bulk modulus with respect to pressure. The internal energy is found by integration:

$$E(V) = E_0 + \frac{9V_0B_0}{16} \left\{ \left[\left(\frac{V_0}{V} \right)^{2/3} - 1 \right]^3 B_1 + \left[\left(\frac{V_0}{V} \right)^{2/3} - 1 \right]^2 \left[6 - 4 \left(\frac{V_0}{V} \right)^{2/3} \right] \right\}. \quad (16)$$

The targets of this test are the three physically measurable parameters: V_0 , B_0 and B_1 . By comparing the predictions of these parameters for the three potentials we can select the one that performs the best among the three. We can also assess the performance of the UPET potential family as a whole against earlier iterations of machine-learned potentials.

The reference dataset of EOS parameters was generated with two all-electron codes (FLEUR [29] and WIEN2K [30]) and averaged. For chemical elements 1-96, it includes 4 mono-element/unary cubic structures (face-centred cubic, body-centred cubic, simple cubic and diamond) and 6 binary cubic oxides (X_2O , XO , X_2O_3 , XO_2 , X_2O_5 , XO_3) [28]. As OAM and OMAD only contain 8 of those elements in their training datasets, the other 7 were

excluded from the experiment. For each of the remaining 890 structures and 7 equidistant volumes from 94% to 106% of a predetermined central volume $\tilde{V}_0$, total energy was calculated by the selected MLIPs in the Atomic Simulation Environment. The equation of state is then fit to the data.

It is worth noting that a number of the structures in the dataset are unstable by nature, so MLIPs would not be able to predict energy or fit the EOS to all of them. In this matter, OAM-L turned out to be the most selective: out of 534 oxides it was able to give an EOS fit for 506, and out of 356 unaries to 333. None of the MLIPs in this experiment were able to predict the EOS parameters for all structures in the dataset, and the structures with missing results vary by potential, so for the unconverged structures we decided to set an arbitrary value: 0.0 for each of the EOS parameters, which will result in a 100% relative error. The final count ended up being 89 elements out of 96, 534 oxides and 356 unaries. The elements present in the final dataset can be found in Fig. 3. The comparison of the reference dataset parameters and the predictions of UPET potentials is presented below:

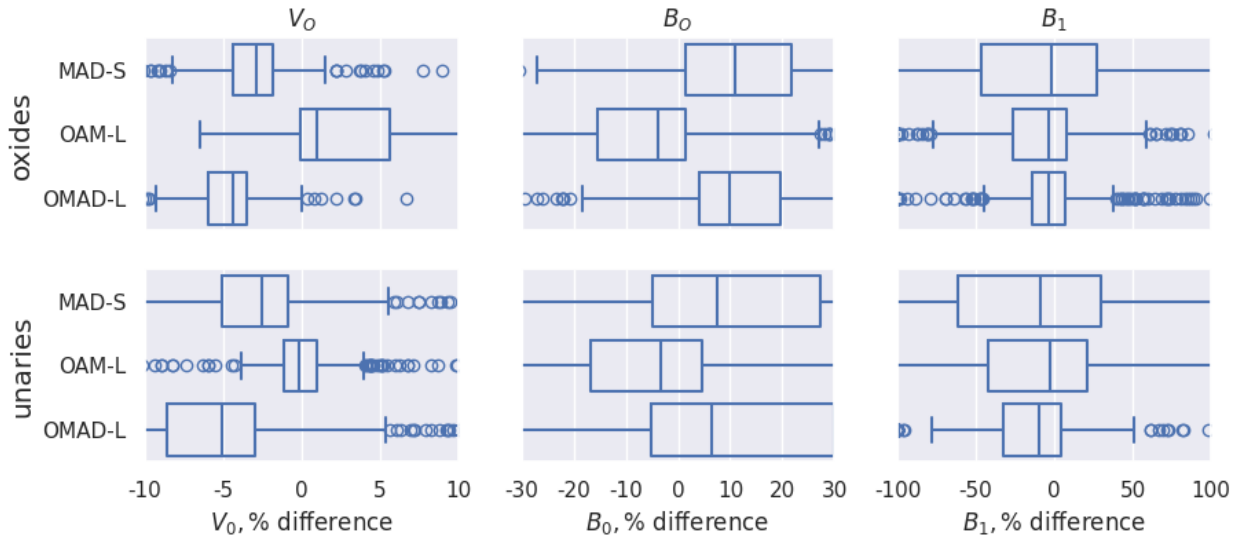


Figure 2: Boxplot showing the relative error in calculations of EOS parameters between the selected UPET family potentials and the reference all-electron dataset

Generally, OAM-L shows a lower skew of the median, and in three out of six subplots it has a smaller inter-quantile range (IQR). It does seem to perform worse with oxides than with unaries, and on average gives worse results for B_1 than OMAD, however the overall smaller skew and significantly better IQRs for V_0 and B_0 still make it the best out of the three.

This conclusion is not surprising: OAM is one of the best-performing potentials for this task on Matbench Discovery and it is trained on three large datasets as compared to one for MAD and OMAD. Aside from that, the developers of the UPET package recommend

OAM for use in material discovery tasks, which include computing convex hull energies and performing structural relaxation.

The quality of prediction varies notably by chemical element and structure, but usually if the EOS managed to converge at all, it reflects the energy around minimum reference volume well. We can quantify the quality of prediction with the metric ν , representing the relative deviation for the combination of the three parameters between two computational approaches:

$$\nu_{w_{V_0}, w_{B_0}, w_{B_1}}(a, b) = 100 \sqrt{\sum_{Y=V_0, B_0, B_1} w_Y \left[\frac{Y_a - Y_b}{(Y_a + Y_b)/2} \right]^2}, \quad (17)$$

where a and b are different computational approaches, that is MLIP-ASE and the parameters in the reference dataset, $w_{V_0} = 1, w_{B_0} = 1/20, w_{B_1} = 1/400$ are weights [28]. For the PET-OAM-L potential, the average ν per chemical element is presented below:

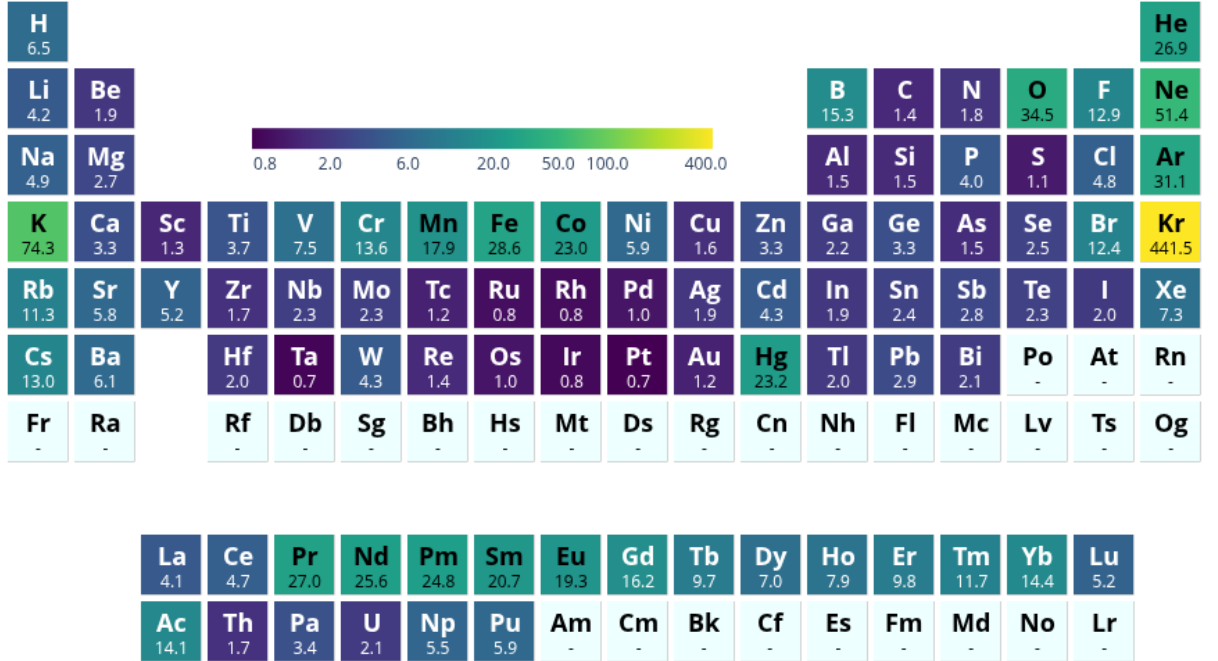


Figure 3: Average value of ν for the converged structures of the chemical elements in the dataset, for PET-OAM-L. Plot created with pymatviz [1]

As we can see, the potential performs the best on the heavier transition metals and, with the exception of noble gases, the p-block of the periodic table. That is where the error is lowest and where we see the least convergence errors. By contrast, the noble gases cause the highest error and convergence failures:

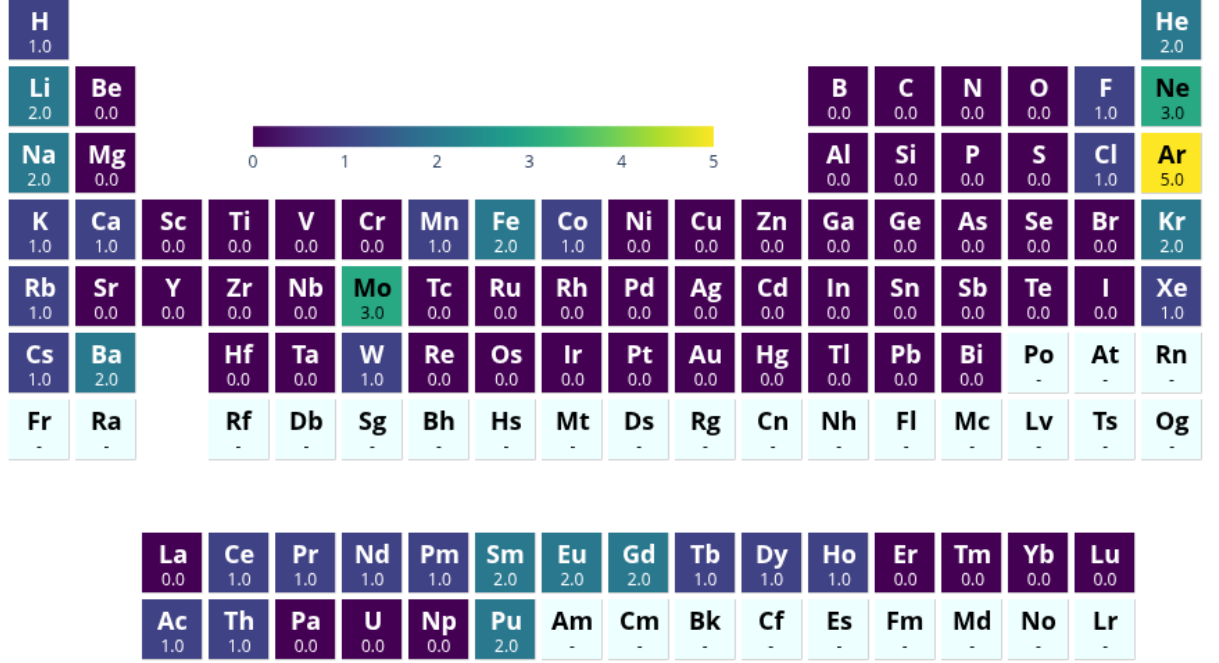


Figure 4: Number of structures with failures of convergence of the EOS, for PET-OAM-L.
Plot created with pymatviz [1]

These results for PET-MAD-S and PET-OMAD-L can be found in figures 13 and 14 in Appendix A. By comparison, they have a higher number of converged structures and a lower maximum ν value, however they cannot achieve the precision OAM-L shows for certain elements in the d- and p-blocks.

The image below visualises the way ν reflects the prediction of the EOS, the four plots represent very different values of the metric: the minimum value achieved by OAM-L with ThO_2 , the median value with the diamond configuration of Cu, maximum value among converged structures with KrO_2 and a simple cubic structure of Sm with an unconverged EOS. E_{ASE} represents the seven energy values we got directly from ASE in relation to the minimum among them, EOS_{Pred} is the fit of the EOS curve, EOS_{True} is the curve with the parameters in the reference dataset:

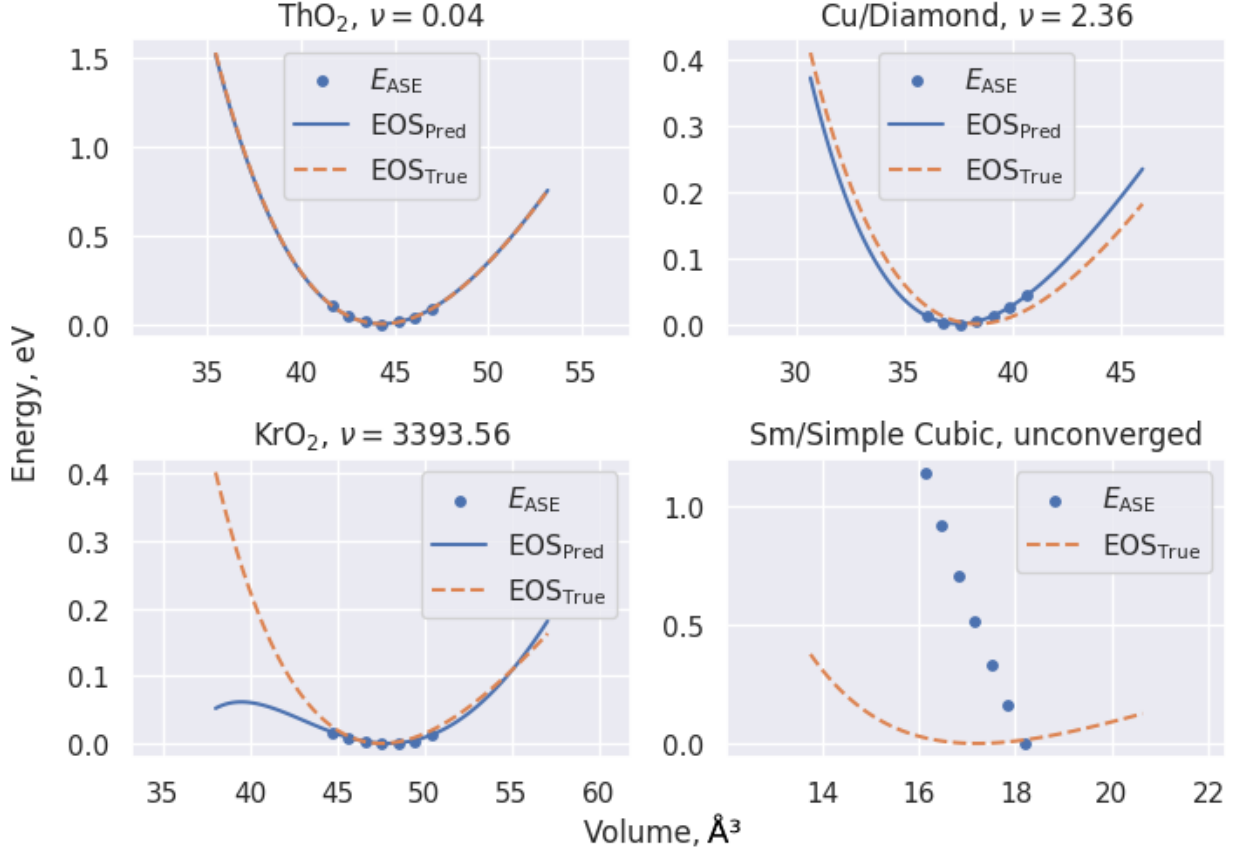


Figure 5: Representations of different values of ν in the calculations of EOS parameters with PET-OAM-L

As we can see, in the first case the predicted EOS curve perfectly reproduces the reference curve, in the second there is a visible deviation, though the value of equilibrium volume is still close, the general shape is kept and the error on energy is relatively small near minimum. In the case of KrO_2 , the curve is reproduced almost to perfection near minimum and to the right, however the shape of the curve is completely broken: the fit resulted in a negative value of B_1 (derivative of the bulk modulus), which is unreasonable and thus the curve has no physical meaning. Finally, on the last curve there is no convergence at all: had there been more points in the experiment or had they been more widely spaced, we would have been able to get some fit, but it is clear the results would have a big margin of error regardless.

It is also interesting to compare these results with earlier GNN-based universal MLIPs. A study done in 2024 [2] assessed four of them: CHGNet, M3GNet, MACE and ALIGNN. The median volume V_0 is predicted accurately in all, however the variance shown by the older potentials is worse: the IQR then was 8-15%, as compared to 2-6% in the MLIPs of the UPET family. The bulk modulus and its derivative show an even starker improvement: in 2024 the IQR for predictions of B_0 and B_1 was over 50 and 80% respectively on all potentials, while among those we selected MAD-S showed the worst result with 31% on B_0

and 90% on B_1 , while OAM-L had IQRs as low as 21 and 63% respectively. The errors decrease further if we take out the structures with the unconverged EOS from the dataset; for example for OAM-L the errors for B_0 and B_1 would decrease to 13 and 46%.

This boxplot, presented in the study [2], demonstrates the evaluation of those potentials:

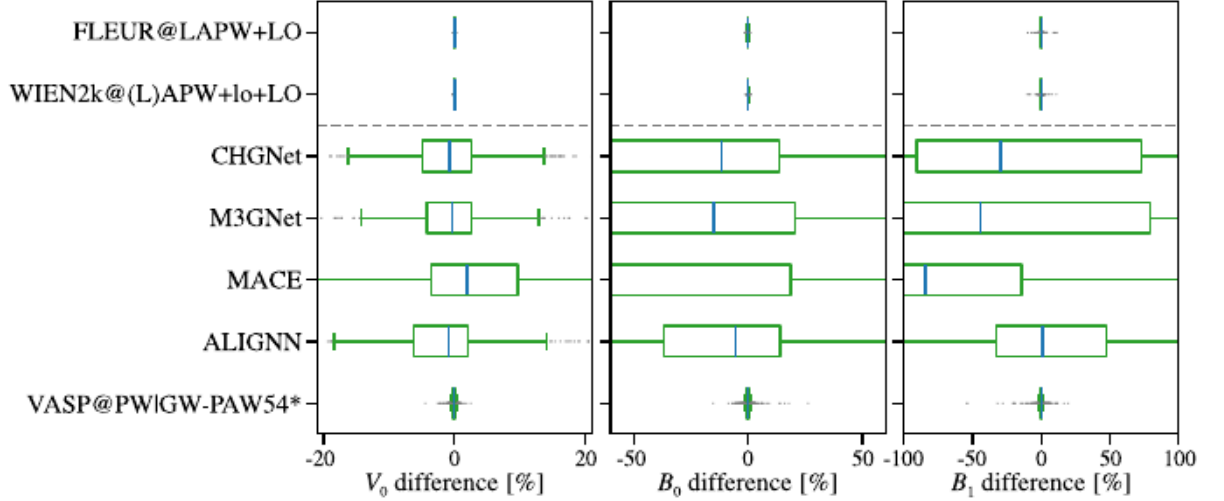


Figure 6: Boxplot showing the relative error in calculations of EOS parameters between the older GNN-based potentials and the reference all-electron results, from the study [2].

Reused under CC BY 4.0

This demonstrates a clear improvement in the quality of predictions over the past two years, as well as the clear choice for future testing: OAM-L, as the best-performing of the three. Now we need to see how it compares with ABINIT and its own reference exchange-correlation functional: OAM-L is trained on PBE data, which is available for use in ABINIT via a `psp8` format pseudopotential [31], found on PseudoDojo [32]. This way we can verify both the correctness of UPET with respect to its training dataset and check that the difference in DFT computation code does not affect the quality of the final results.

We have decided to use the method of parity plots, outlined in [33]. We have taken the diamond configuration of silicon (structure mp-149 [34] on Materials Project [35]), and perturbed it by a random vector using AbiPy ($\|\Delta\mathbf{r}\| \leq 0.1 \text{ \AA}$). For these 15 perturbed structures, we calculated their energies, as well as the forces and stresses acting on the atoms, both in ABINIT with the PBE-based pseudopotential (SCF convergence criterion: `toldfe` = 10^{-6} Ha), and with OAM-L in the Atomic Simulation Environment (ASE). The results are presented below, along with the mean absolute error (MAE) of each category:

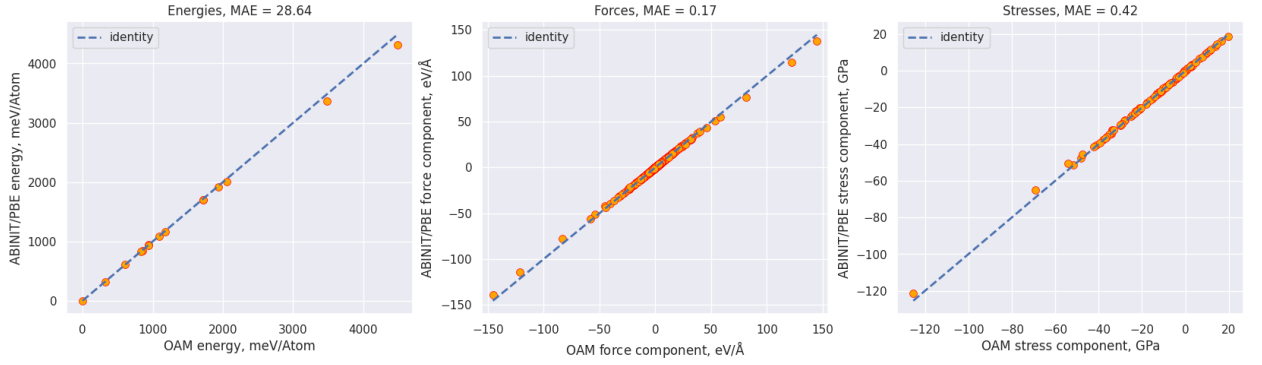


Figure 7: Parity plots of forces, energies and stresses of 15 perturbed diamond silicon configurations, between ABINIT-PBE and PET-OAM-L calculations

The results in calculations done with OAM correspond well with the calculations in ABINIT, no matter how far they are from the energy minimum. The mean absolute error also proves to be comparatively small. This means that, should UPET prove to be consistently faster in calculations, we can use it to get a structure close to the true energy minimum and save a lot of time and computational resources in the process. We will also test the convergence from the intermediate structure to the minimum energy configuration in ABINIT.

III.2 Efficiency of UPET in structure relaxation

Early tests showed a large decrease in time required for a step of the BFGS optimisation algorithm when using OAM-L as opposed to ABINIT. We wanted to measure the observed speed-up, on both a personal computer and a computational cluster, as well as the number of BFGS steps required to optimise a structure via these methods.

The computational cluster used for these tests was Lemaitre4, a massively parallel cluster hosted at UCLouvain that utilises the SLURM architecture. For both a PC and Lemaitre4, the CPU was utilised as the computational component: as both ABINIT and UPET support GPU computing, neither has an advantage in this regard. GPU computing is also generally less accessible.

Characteristic	PC	Lemaitre4 cluster
Processor	AMD Ryzen 5 7535HS	AMD Epyc Genoa 9534
Cores available	6	5120
Cores in ABINIT	1	1-12
Clock speed	3.3 GHz	3.7 GHz
L3 Cache	16 MB	256 MB

Table 1: Characteristics of the computational devices used in the study

A 16-atom supercell of silicon in the diamond configuration was generated and perturbed by a random vector ($\|\Delta\mathbf{r}\| \leq 0.1 \text{ \AA}$) in AbiPy. For 50 perturbed structures, we conducted geometry optimisation in ABINIT using a PBE-based pseudopotential and the BFGS algorithm for the atomic configuration. The SCF convergence criterion was set as `toldfe` = 10^{-6} Ha, while the BFGS convergence criterion was set as `tolmx` = $5 \cdot 10^{-5}$ Ha/Bohr = $0.0026 \text{ eV/\AA}$. The results are presented below:

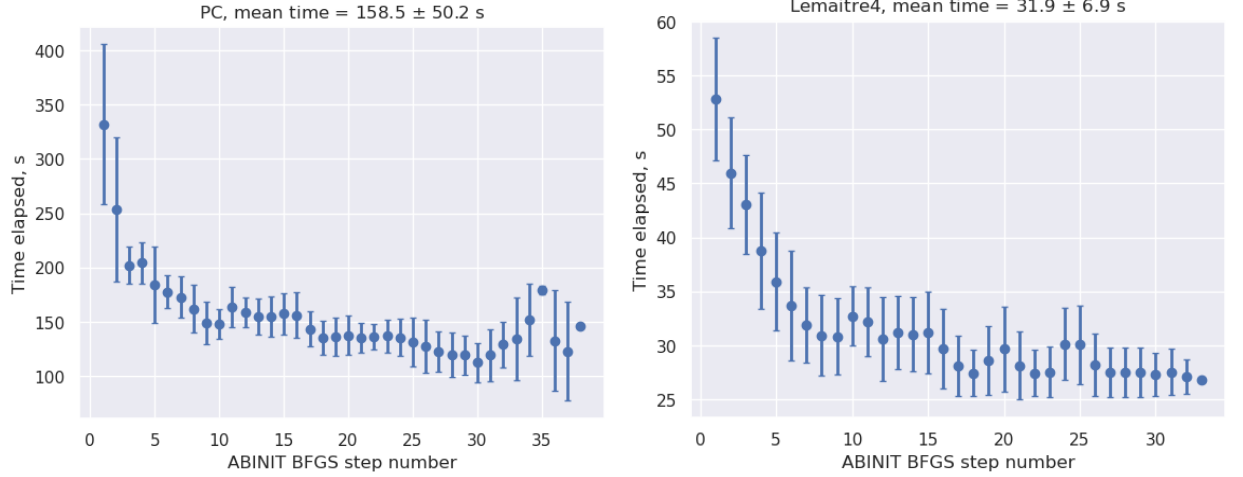


Figure 8: Time required for a BFGS step in ABINIT for a perturbed 16-atom diamond Si supercell. Calculations performed on a) a personal computer and b) cluster Lemaitre4

As expected, the cluster performs the ABINIT calculations much faster than a personal computer, with the mean time on a PC being five times larger than that on the cluster. However, it is also interesting to note higher times required during early optimisation steps: higher forces acting on atoms require more SCF steps for energy convergence. Later in the simulation the average count of SCF steps needed lowers until reaching a plateau, and so does the time required to reach the needed tolerance.

A similar experiment was done with OAM-L. Using the `UPETCalculator` method with ASE and the BFGS algorithm (max force tolerance = $0.005 \text{ eV/\AA}$), we performed structural relaxation on the same 50 perturbed structures, measuring the time required for a BFGS step.

Calculator	Step time on PC, s	Step time on Lemaitre4, s
ABINIT-PBE	158.5 ± 50.2	31.9 ± 6.9
PET-OAM-L	0.23 ± 0.02	0.81 ± 0.02

Table 2: Average time required for a step of the BFGS optimisation algorithm in the 16-atom diamond Si experiment, on PC and Lemaitre4

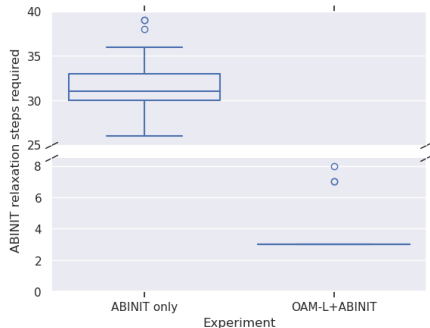
OAM-L, unlike ABINIT, does not show a direct correlation between the BFGS step

number, as it does not employ iterative methods in its predictions. Comparing the averages, we can see an average speed-up ($t_{\text{UPET}}/t_{\text{DFT}}$ of 40 on the cluster and of 700 on the PC per step. Granted, the figure for the PC may be overestimated: on a cluster UPET calculations are forced to use 1-2 cores due to the way SLURM task distribution works, while on a PC there is no such restriction, and the processor may optimise calculations accounting for it. Even if we consider the UPET calculation time to be roughly equivalent on both, the speed-up observed on the PC would be about 200, which is more impactful than on Lemaitre4, where it is already large.

In this experiment, it took 30.2 ± 1.7 steps on average to reach a force tolerance of $0.0026 \text{ eV/\AA}$ on a perturbed structure, while it took UPET 28.8 ± 4.5 steps on average to reach a tolerance of $0.005 \text{ eV/\AA}$, which in terms of time required would equate to 0.73 ABINIT steps on the cluster, or 0.04 steps on the PC. These measurements render any time spent for structure optimisation with UPET negligible compared to the DFT calculations.

If structural relaxation done in UPET is able to lower the number of BFGS steps we need to do in ABINIT, we will be able to save computational resources while still getting the benefit of higher accuracy provided by DFT methods. An additional resource-saving factor comes from the shape of the time curve: by skipping the initial structural optimisation steps, we skip more SCF cycles than average, thus reducing the time even further.

In the following experiment, we conducted geometry optimisation on a different set of 50 perturbed structures, based on the same 16-atom supercell of diamond Si with the same perturbation characteristics. For each structure, optimisation was conducted both with ABINIT only and with a combination of OAM-L and ABINIT, with the UPET force maximum set to $0.005 \text{ eV/\AA}$. For each of these optimisations, we noted the number of BFGS steps required to reach the force tolerance of $0.0026 \text{ eV/\AA}$ in ABINIT. The results are presented below:



	ABINIT	OAM-L+ABINIT
Steps	31.6 ± 2.7	3.2 ± 1.0

Figure 9: Effect of PET-OAM-L on the number of BFGS relaxation steps in ABINIT in the 16-atom experiment on Si

As we can see, PET-OAM-L significantly decreases the number of BFGS steps and

drastically lowers computation time. We can see a number of outliers requiring 7-8 steps to reach the minimum energy configuration: in those cases, the error accumulated in the use of a machine-learning potential resulted in the optimisation going towards a local minimum instead of the global, resulting in a correction within DFT and a larger number of steps required. Even in this worst case, roughly 12 minutes of machine-time were conserved, while for the majority of calculations it would be 15 minutes.

It would be reasonable to assume that the amount of computational resources conserved rises with a higher atomic count: standard DFT methods scale in a cubic manner with the size of the system, and despite all the optimisations done to the computation process that scaling should still outstrip that of a machine-learning potential. To test this theory, a similar experiment was conducted on a 54-atom supercell of diamond silicon, with 10 perturbed structures. The calculations were done on Lemaitre4 with the same SCF and BFGS convergence criteria, as in the previous experiment:

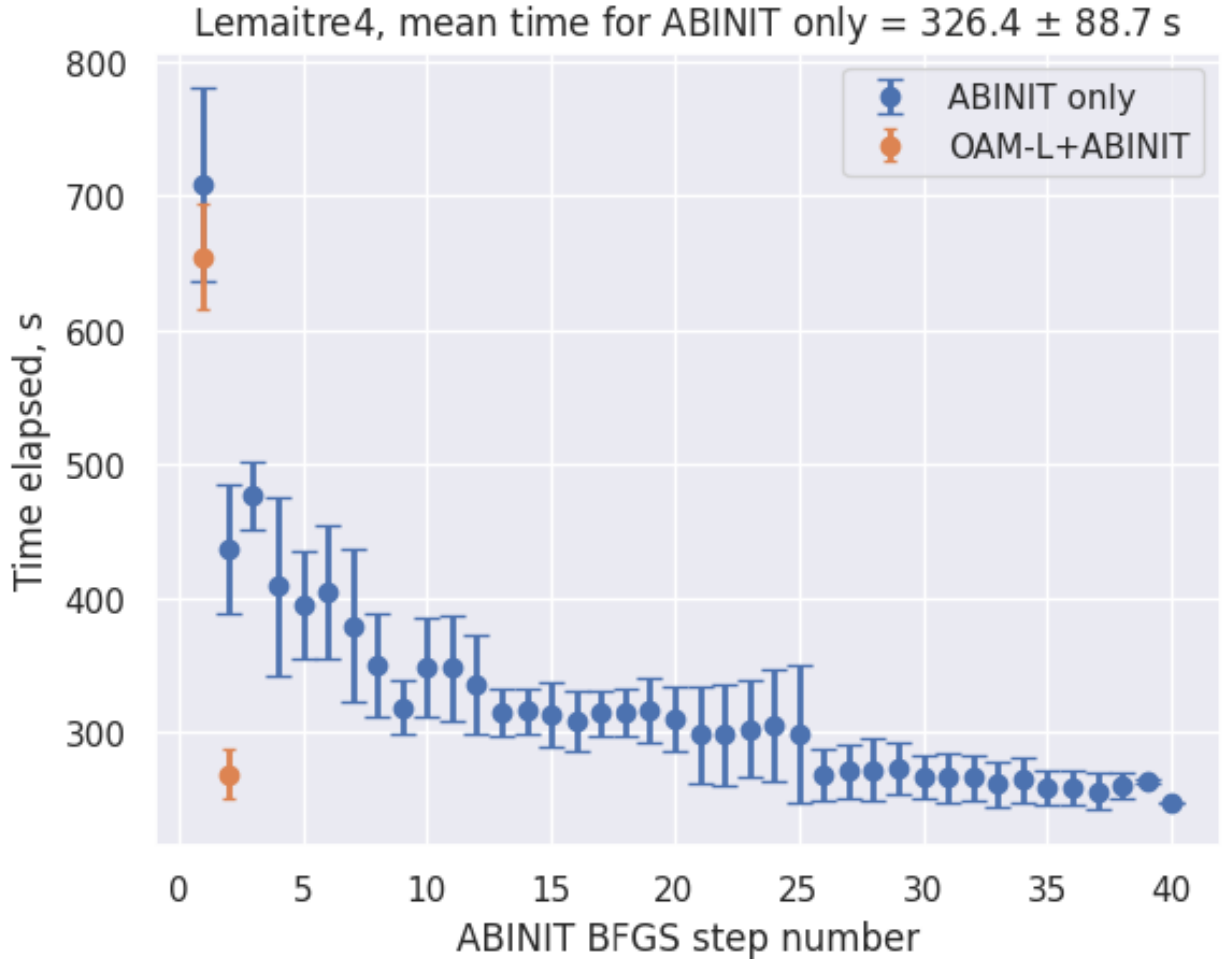


Figure 10: Time required for a BFGS step in ABINIT for a perturbed 54-atom diamond Si supercell, with and without prior relaxation with OAM-L

Framework	ABINIT	OAM-L+ABINIT
Average time per step, s	326.5 ± 88.7	1.7 ± 0.1
BFGS steps with OAM-L	–	30.6 ± 1.5
BFGS steps with ABINIT	37.6 ± 1.4	2.0 ± 0.0

Table 3: Effect of PET-OAM-L on the relaxation of the perturbed 54-atom Si supercell in ABINIT

Under the same conditions, the average BFGS step time in ABINIT increased by over 10 times, while the step time of OAM only doubled. Meanwhile, the number of steps needed to reach the global minimum increased for ABINIT-only calculations, but stayed about the same for the structures pre-relaxed by the machine-learning potential. The observed speed-up for this task is roughly 200, and on average about 3.2 cluster machine-hours would be saved per structure. This is already a significant improvement that will no doubt scale further as the number of atoms is increased. And, of course, once DFT calculations become completely unaffordable, those utilising UPET would still take reasonable time and produce sensible results.

It is worth noting that ABINIT and OAM take different optimisation paths which result in different Cartesian coordinates, nonetheless following the same symmetry:

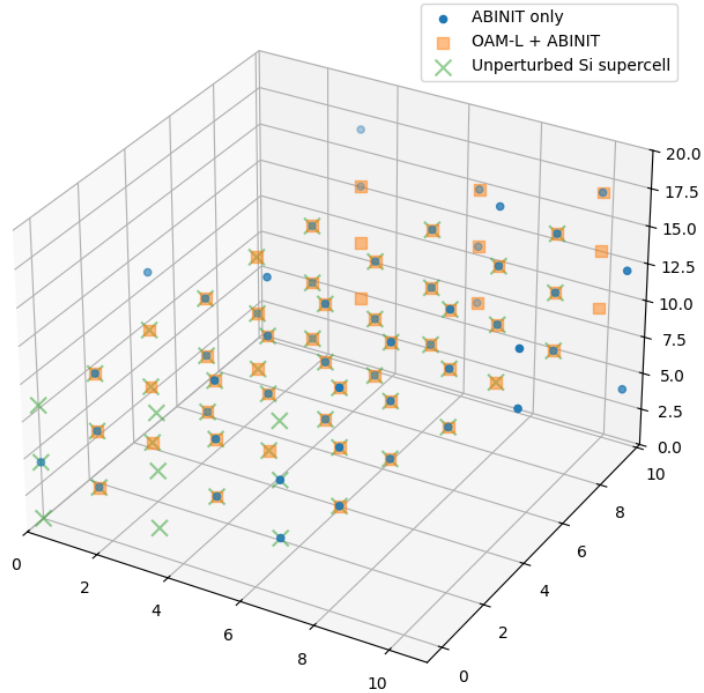


Figure 11: Three-dimensional view of three diamond Si supercells: the unperturbed configuration, one optimised only with ABINIT and one optimised with OAM and ABINIT

Finally, we have decided to test the performance of OAM-L on a more complex system, specifically the orthorhombic crystal of P_2O_5 (structure mp-2173 on Materials Project [36]). The presence of two atomic species, a difference in electronegativity, and a generally more complex geometry should make this system more difficult to compute for both DFT and ML methods. OAM-L was able to successfully evaluate this oxide in III.1, so it was chosen for this test.

The conventional cell of this structure contains 28 atoms, 8 of which are phosphorus and 20 are oxygen; this number is sufficient for our test. 50 randomly perturbed structures ($(\|\Delta\mathbf{r}\| \leq 0.1 \text{ \AA})$) were generated, the SCF convergence criterion was set as `tolvrs` = 10^{-12} , while the BFGS convergence criterion remained the same at `tolmxf` = $0.0026 \text{ eV/\AA}$. As in the previous experiment, they were optimised both only with ABINIT and with a combination of OAM-L and ABINIT, with the OAM-L force convergence criterion set at $0.005 \text{ eV/\AA}$. The results of the experiment can be seen below:

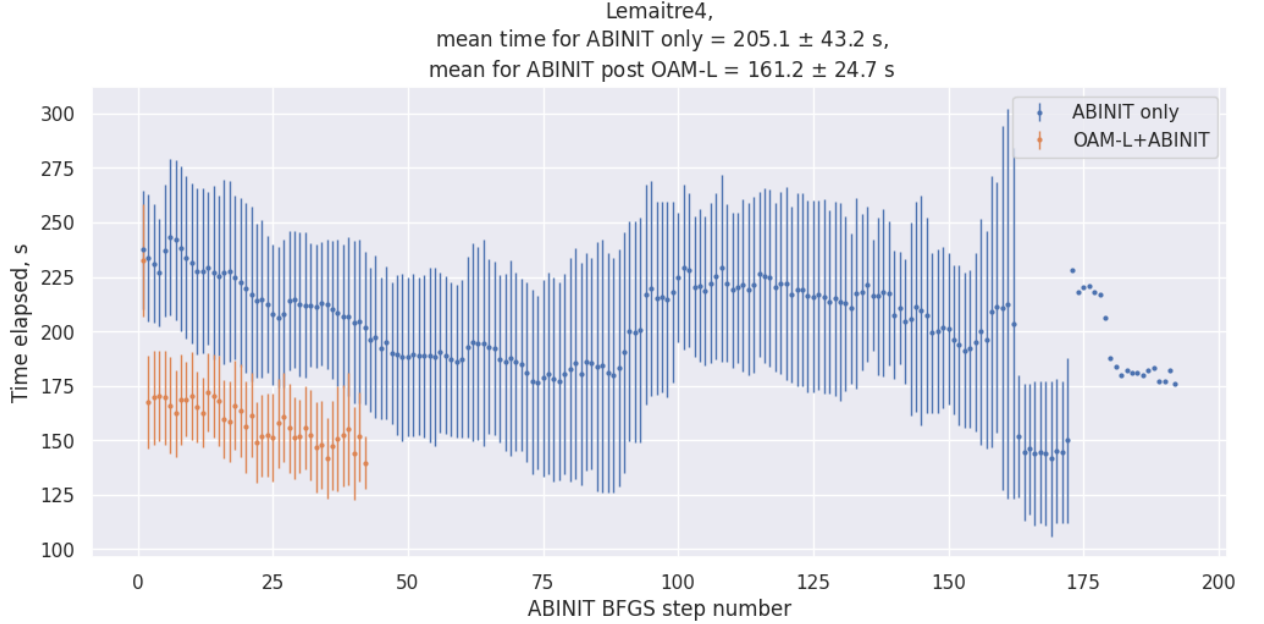


Figure 12: Time required for a BFGS step in ABINIT for a perturbed P_2O_5 cell, with and without prior relaxation with OAM-L

Framework	ABINIT	OAM-L+ABINIT
Average time per step, s	193.3 ± 43.7	1.36 ± 0.01
BFGS steps with OAM-L	—	56.7 ± 3.3
BFGS steps in ABINIT	103.7 ± 32.3	38.1 ± 4.7

Table 4: Effect of PET-OAM-L on the relaxation of the perturbed conventional cell of P_2O_5 in ABINIT

We note both a significant reduction in step count, and the reduction of time/SCF cycles required for a single optimisation step. We can also see that OAM-L helps mitigate difficulty with optimisation of problematic configurations: 35 out of 50 structures converged in the ABINIT-only test in under 100, most of them under 90, while the other fifteen required 118+ steps, with the highest being 192 (this also explains the rise in average time for the ABINIT-only test around step number 100). However, in the test with OAM-L optimisation, all of the structures reached the global minimum within 31-42 steps, with some among the problematic fifteen on the lower end of that range. On average, using OAM-L for relaxation in this case saves 4.2 cluster machine-hours per structure.

We have demonstrated that for a variety of systems of different size, composition and chemical properties, OAM-L significantly accelerates structural relaxation in comparison to a DFT-based method in ABINIT. As both computational frameworks are based on the PBE functional, they can be used together, which ultimately allows us to obtain DFT-accurate results at a lower computational cost through the use of the machine-learning potential.

IV Implementation considerations

All calculations performed in the previous chapter were controlled in Python, via the AbiPy interface for ABINIT and the Atomic Simulation Environment (ASE) for UPET. Theoretically, this should be simple: ASE itself provides options to interact with ABINIT, however this link is prone to breaking in major updates, and is difficult to configure, as ABINIT is ultimately a Fortran-based computational framework, with Python code present only for interface purposes. In fact, arranging these calculations necessitated juggling multiple atomic representations, varied file output and format conversions. To an untrained eye, it is also unclear whether ASE methods are able to function inside AbiPy workflows. All of these can complicate writing and debugging simulation code, as well as waste time dealing with errors that inevitably arise from complex program configurations. So, it would be quite useful to be able to utilise UPET potentials directly within AbiPy, or even through ABINIT scripts if possible.

The primary issue with this approach is the size of the models themselves. To properly represent a physical system, neural networks have to comprise many layers and learnable parameters, whose number is high even for the base PET architecture, and was raised even higher for UPET [9]. The model primarily used in the study, OAM-L has 255 million parameters, translating to 772 MB of physical memory. For the state-of-the-art potential reflected on Materials Discovery, OAM-XL, those numbers increase to 730 million parameters and 2.92 GB of memory, respectively. For comparison, the source code of ABINIT in version 10.8.3 takes up only 163 MB. As a result, building UPET models directly into ABINIT or AbiPy is irrational. Including them as configuration options during program setup is a more reasonable alternative, but there is another approach, already used in ASE.

UPET potentials are primarily distributed as a Python package with the same name, which provides an ASE-compatible `UPETCalculator`, which handles energy, force and stress calculations. This calculator is then passed to a structural optimiser (such as `BFGS`) or another algorithm to provide necessary data in the study. Notably, ASE supports an interface: UPET, other MLIPs, even ABINIT are all integrated with the package as calculators, which only take atomic configuration and handle all actual computation inside. For ASE to function we do not need to have UPET installed, and only when we want to utilise these potentials we have to install the `upet` package and download the potentials we need by invoking them in the code.

AbiPy provides support for some ASE methods and MLIPs within its framework, though the `abipy.ml.aseml` module. The module itself essentially acts as an interface between AbiPy and ASE: the methods in use are directly taken from ASE, it can load ASE calculators and utilise them in studies. In a way, the problem of compatibility for Python is already

solved, except that most of the problems regarding data types and format conversion persist, only having been moved inside AbiPy instead. For convenience purposes, it may be worth adding some of the modern UPET potentials as options for the `CalcBuilder` class, which allows AbiPy to call the potentials by name and integrates unique corrections. PET-MAD already has an interface through it, a suggested addition integrating OAM-XL can be found on page 41.

Interestingly, UPET supports a wider range of atomistic frameworks: alongside ASE, it supports LAMMPS [37], GROMACS [38] and three other simulation engines. In the case of LAMMPS, the setup is slightly more involved than usual, however the potentials can be used freely with full compatibility: when using the correct software configuration, LAMMPS will accept `metatomic` as a pair style, which governs the calculations.

The UPET family of potentials is not the only one to be portable in this manner. A number of TorchScript (PyTorch [39])-based models, such as the Gaussian Approximation Potential [40], are also available for use through the shared `metatomic` interface, which enables the exchange of code between different packages [41]. The `metatensor` package that is distributed alongside it allows for the exchange of data, for example stored gradients and their descriptive information, and has methods for efficient storage of sparse matrices. Combined, these libraries aim to improve the interoperability and usability of ML software with various computational frameworks; and considering that every model compatible with TorchScript (like GNNs written in PyTorch) can be integrated with them, they represent a strong promise for the future of atomistic simulation.

It would be very beneficial to implement support for these interfaces in ABINIT, as it would give native access to the existing and future models, likely to be compatible with this interface. The newest models represented on Matbench Discovery are all built in PyTorch, thus being fully compatible with the interface. Most programming languages (including Fortran) are able to call an Application Programming Interface (API) in C, through which `metatomic` provides access to C++, Python and Rust, creating a number of implementation options. Implementing the `metatomic` interface in ABINIT could be a goal for future development.

Another interesting perspective involves the approximate Hessian from UPET calculations. As discussed in II.2, we begin the calculations with some initial estimation or the identity matrix, and gradually update the Hessian in the BFGS optimisation algorithm. However, if we were to do an initial optimisation with UPET, we would already have a Hessian somewhat tailored to the system, and it would likely be closer to the true Hessian than other initialisations.

It has been shown in older studies that the utilisation of an estimated Hessian using the structure motif decomposition [42] or the Lennard-Jones potential [43] leads to a speed-up of

2-5 times. In our case, after reaching a fairly low tolerance on maximum force, it should be a matrix closer to the real state of the system, with a likely further reduction in SCF steps required. In ABINIT, the variable controlling the initial matrix is `iprcfc`, and currently it only supports fractions or multiples of the identity matrix. If we were able to pass the ASE-format `.json` Hessian file, or some other format, no doubt that could speed up the computation further.

V Conclusion

In this work we have proposed a machine-learning method for accelerating structural relaxation. UPET, the selected family of machine-learning interatomic potentials, was assessed in terms of the accuracy of calculation of energy, acting forces and stresses, as well as physical properties of structures. The best performing potential in the selection, PET-OAM, was then used in structural relaxation, both on its own and in combination with DFT methods. The experiments have demonstrated a significant acceleration of calculations while maintaining the accuracy of DFT methods. Finally, the implementation of this potential in the ABINIT computational framework was discussed.

Acknowledgements

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

References

- [1] J. Riebesell, H. Yang, R. Goodall, and S. G. Baird, “Pymatviz: visualization toolkit for materials informatics,” 2022. [Online]. Available: <https://doi.org/10.5281/zenodo.7486816>
- [2] H. Yu, M. Giantomassi, G. Materzanini, J. Wang, and G.-M. Rignanese, “Systematic assessment of various universal machine-learning interatomic potentials,” *Materials Genome Engineering Advances*, vol. 2, no. 3, p. e58, 2024. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/mgea.58>
- [3] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, 01 2001, vol. 1, p. 664.
- [4] D. Sholl and J. A. Steckel, *What is Density Functional Theory?* John Wiley Sons, Ltd, 2009, ch. 1, pp. 1–33. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470447710.ch1>
- [5] I. Goddard, William A., “Classical force fields and methods of molecular dynamics,” in *Computational Materials, Chemistry, and Biochemistry: From Bold Initiatives to the Last Mile*. Springer International Publishing, Jan. 2021, pp. 1063–1072. [Online]. Available: https://doi.org/10.1007/978-3-030-18778-1_45
- [6] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csanyi, A. V. Shapeev, A. P. Thompson, M. A. Wood, and S. P. Ong, “Performance and cost assessment of machine learning interatomic potentials,” *The Journal of Physical Chemistry A*, vol. 124, no. 4, pp. 731–745, 01 2020. [Online]. Available: <https://doi.org/10.1021/acs.jpca.9b08723>
- [7] X. Gonze *et al.*, “The Abinit project: Impact, environment and recent developments,” *Computer Physics Communications*, vol. 248, p. 107042, 2020. [Online]. Available: <https://doi.org/10.1016/j.cpc.2019.107042>
- [8] A. H. Larsen *et al.*, “The atomic simulation environment—a Python library for working with atoms,” *Journal of Physics: Condensed Matter*, vol. 29, no. 27, p. 273002, 2017. [Online]. Available: <https://doi.org/10.1088/1361-648X/aa680e>
- [9] F. Bigi, P. Pegolo, A. Mazitov, and M. Ceriotti, “Pushing the limits of unconstrained machine-learned interatomic potentials,” 2026. [Online]. Available: <https://arxiv.org/abs/2601.16195>
- [10] W. Kohn and L. J. Sham, “Self-Consistent Equations Including Exchange and Correlation Effects,” *Physical Review*, vol. 140, pp. A1133–A1138, 1965. [Online]. Available: <http://dx.doi.org/10.1103/PhysRev.140.A1133>

- [11] D. Sholl and J. A. Steckel, *Accuracy and Methods beyond “Standard” Calculations*. John Wiley Sons, Ltd, 2009, ch. 10, pp. 209–233. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470447710.ch10>
- [12] J. P. Perdew, K. Burke, and M. Ernzerhof, “Generalized Gradient Approximation Made Simple,” *Phys. Rev. Lett.*, vol. 77, pp. 3865–3868, Oct 1996. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.77.3865>
- [13] S. Shaik, I. S. Edward, A. S. Robert, and R. B. King, “Computational Inorganic and Bioinorganic Chemistry,” *Angewandte Chemie International Edition*, vol. 49, no. 18, pp. 3111–3113, 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201000436>
- [14] F. Jensen, *Introduction to Computational Chemistry – 2nd ed.* John Wiley Sons, Ltd, 2007, ch. 12, pp. 380–419. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470447710.ch10>
- [15] R. Fletcher, *Newton-Like Methods*. John Wiley Sons, Ltd, 2000, ch. 3, pp. 44–79. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118723203.ch3>
- [16] J. Behler, “Perspective: Machine learning potentials for atomistic simulations,” *The Journal of Chemical Physics*, vol. 145, no. 17, p. 170901, 11 2016. [Online]. Available: <https://doi.org/10.1063/1.4966192>
- [17] E. Prodan and W. Kohn, “Nearsightedness of electronic matter,” *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, pp. 11 635–11 638, 2005. [Online]. Available: <https://www.pnas.org/doi/abs/10.1073/pnas.0505436102>
- [18] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges,” 2021. [Online]. Available: <https://arxiv.org/abs/2104.13478>
- [19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The Graph Neural Network Model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
- [20] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural Message Passing for Quantum Chemistry,” 2017. [Online]. Available: <https://arxiv.org/abs/1704.01212>
- [21] J. Riebesell, R. E. A. Goodall, P. Benner, Y. Chiang, B. Deng, G. Ceder, M. Asta, A. A. Lee, A. Jain, and K. A. Persson, “A framework to evaluate machine learning

- crystal stability predictions,” *Nature Machine Intelligence*, vol. 7, no. 6, pp. 836–847, Jun 2025. [Online]. Available: <https://doi.org/10.1038/s42256-025-01055-1>
- [22] S. N. Pozdnyakov and M. Ceriotti, “Smooth, exact rotational symmetrization for deep learning on point clouds,” 2024. [Online]. Available: <https://arxiv.org/abs/2305.19302>
- [23] N. Shazeer, “GLU Variants Improve Transformer,” 2020. [Online]. Available: <https://arxiv.org/abs/2002.05202>
- [24] B. Deng, P. Zhong, K. Jun, J. Riebesell, K. Han, C. J. Bartel, and G. Ceder, “CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling,” *Nature Machine Intelligence*, vol. 5, no. 9, pp. 1031–1041, Sep 2023. [Online]. Available: <https://doi.org/10.1038/s42256-023-00716-3>
- [25] J. Schmidt, T. F. Cerqueira, A. H. Romero, A. Loew, F. Jäger, H.-C. Wang, S. Botti, and M. A. Marques, “Improving machine-learning models in materials science through large datasets,” *Materials Today Physics*, vol. 48, p. 101560, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542529324002360>
- [26] L. Barroso-Luque, M. Shuaibi, X. Fu, B. M. Wood, M. Dzamba, M. Gao, A. Rizvi, C. L. Zitnick, and Z. W. Ulissi, “Open materials 2024 (omat24) inorganic materials dataset and models,” 2026. [Online]. Available: <https://arxiv.org/abs/2410.12771>
- [27] A. Mazitov, S. Chorna, G. Fraux, M. Bercx, G. Pizzi, S. De, and M. Ceriotti, “Massive Atomic Diversity: a compact universal dataset for atomistic machine learning,” *Scientific Data*, vol. 12, no. 1, p. 1857, Nov 2025. [Online]. Available: <https://doi.org/10.1038/s41597-025-06109-y>
- [28] E. Bosoni *et al.*, “How to verify the precision of density-functional-theory implementations via reproducible and universal workflows,” *Nature Reviews Physics*, vol. 6, no. 1, pp. 45–58, 2024. [Online]. Available: <https://doi.org/10.1038/s42254-023-00655-3>
- [29] D. Wortmann *et al.*, “FLEUR,” Zenodo, May 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7576163>
- [30] P. Blaha, K. Schwarz, F. Tran, R. Laskowski, G. K. H. Madsen, and L. D. Marks, “WIEN2k: An APW+lo program for calculating the properties of solids,” *The Journal of Chemical Physics*, vol. 152, no. 7, p. 074101, 2020. [Online]. Available: <https://doi.org/10.1063/1.5143061>

- [31] D. R. Hamann, “Optimized norm-conserving vanderbilt pseudopotentials,” *Phys. Rev. B*, vol. 88, p. 085117, Aug 2013. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.88.085117>
- [32] M. van Setten, M. Giantomassi, E. Bousquet, M. Verstraete, D. Hamann, X. Gonze, and G.-M. Rignanese, “The pseudodojo: Training and grading a 85 element optimized norm-conserving pseudopotential table,” *Computer Physics Communications*, vol. 226, pp. 39–54, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010465518300250>
- [33] A. Hernandez *et al.*, “Fast, accurate, and transferable many-body interatomic potentials by symbolic regression,” *npj Computational Materials*, vol. 5, no. 1, p. 112, 2019. [Online]. Available: <https://doi.org/10.1038/s41524-019-0249-1>
- [34] “Materials Data on Si by Materials Project,” 2020. [Online]. Available: <https://doi.org/10.17188/1190959>
- [35] A. Jain *et al.*, “The Materials Project: A materials genome approach to accelerating materials innovation,” *APL Materials*, vol. 1, no. 1, p. 011002, 2013. [Online]. Available: <http://link.aip.org/link/AMPADS/v1/i1/p011002/s1&Agg=doi>
- [36] “Materials Data on P2O5 by Materials Project,” 2020. [Online]. Available: <https://doi.org/10.17188/1197141>
- [37] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in ’t Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales,” *Comp. Phys. Comm.*, vol. 271, p. 108171, 2022.
- [38] M. Abraham, T. Murtola, R. Schulz, S. Páll, J. Smith, B. Hess, and E. Lindahl, “Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers,” *SoftwareX*, vol. 1, 2015.
- [39] A. Paszke *et al.*, *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., 2019.
- [40] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, “Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons,” *Phys. Rev. Lett.*, vol. 104, p. 136403, Apr 2010. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.104.136403>

- [41] F. Bigi, J. W. Abbott, P. Loche, A. Mazitov, D. Tisi, M. F. Langer, A. Goscinski, P. Pegolo, S. Chong, R. Goswami, P. Febrer, S. Chorna, M. Kellner, M. Ceriotti, and G. Fraux, “metatensor and metatomic: Foundational libraries for interoperable atomistic machine learning,” *The Journal of Chemical Physics*, vol. 164, no. 6, p. 064113, 2026. [Online]. Available: <https://doi.org/10.1063/5.0304911>
- [42] Z. Zhao, L.-W. Wang, and J. Meza, “Motif-based hessian matrix for ab initio geometry optimization of nanostructures,” *Phys. Rev. B*, vol. 73, p. 193309, May 2006. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.73.193309>
- [43] Z. Chen, J. Li, S. Li, and L. Wang, “Approximate hessian for accelerating ab initio structure relaxation by force fitting,” *Phys. Rev. B*, vol. 89, p. 144110, Apr 2014. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.89.144110>

Appendix A: Relevant data for feasibility studies

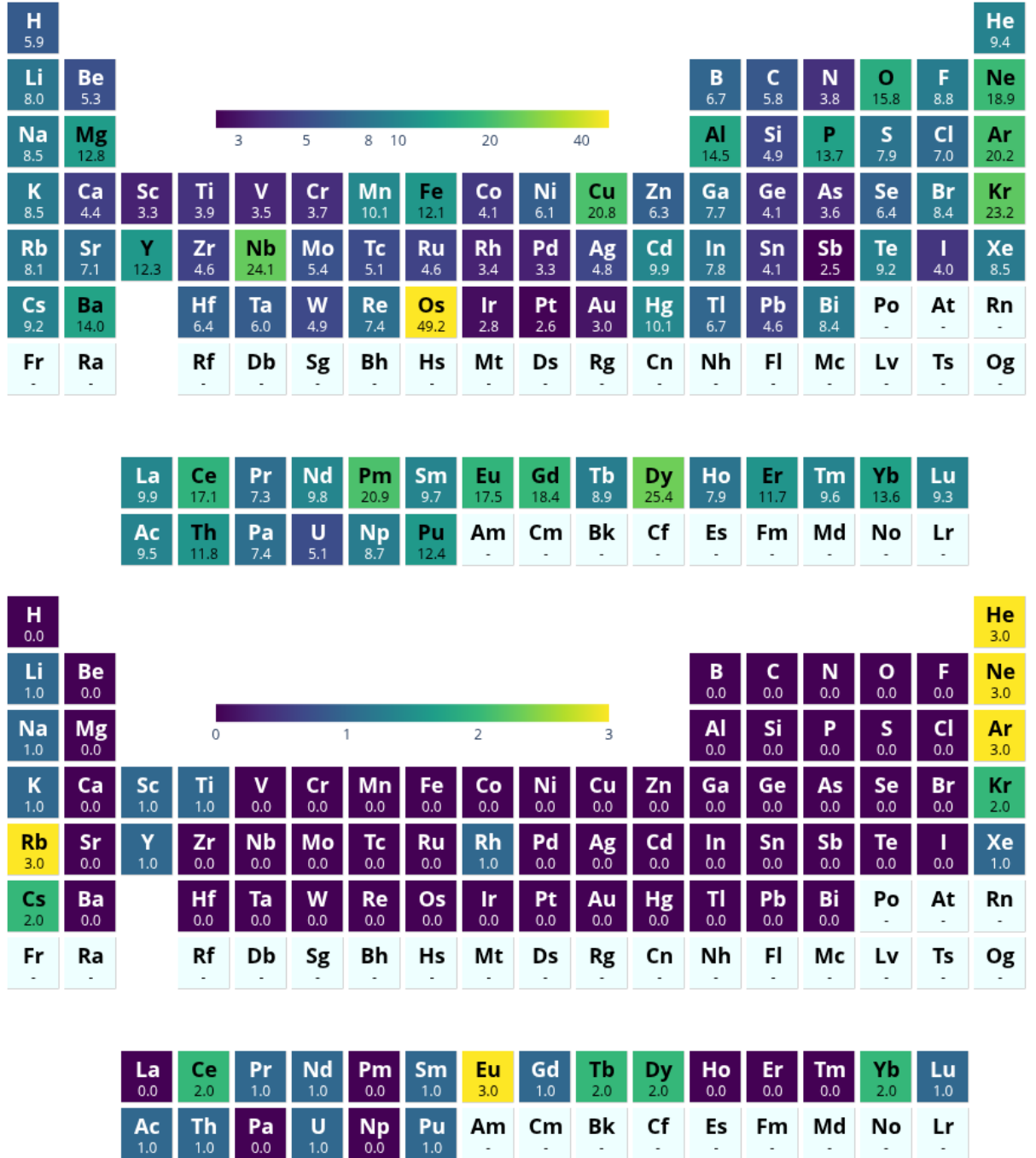


Figure 13: Average value of ν and failures of convergence of EOS by chemical element for PET-MAD-S.

Appendix B: CalcBuilder implementation for PET-OAM-XL

abipy/abipy/ml/aseml.py

Line 1942

```
...
    elif self.nn_type == "pet-oam-xl":
        try:
            from upet.calculator import UPETCalculator
        except ImportError as exc:
            raise ImportError("upet not installed.
            See https://github.com/lab-cosmo/upet") from exc

        import torch
        dev = "cuda" if torch.cuda.is_available() else "cpu"

        class _UPETCalculator(_MyCalculator, UPETCalculator):
            """Add abi_forces and abi_stress"""

        calc = _UPETCalculator(model="pet-oam-xl", device=dev,
                               version="latest")
...

```

Appendix C: ABINIT scripts and Python code used in calculations

ABINIT script used in calculations

```
#### SECTION: basic
ecut 30
toldfe 1e-06 # for Si OR toldvrs 1e-12 for P205
occopt 3
nband 40 # for 16-atom Si, 100 for P205, 130 for 54-atom Si
ngkpt 2 2 2
kptopt 1
nshiftk 1
shiftk 0 0 0
#####
####                      SECTION: dev
#####
iomode 3
optforces 1
ionmov 2
#####
####                      SECTION: files
#####
prtwf 0
prtden 0
prtpot 0
pseudos "Si.psp8"
#####
####                      SECTION: gstate
#####
tsmear 0.01
chkprim 0
#####
####                      SECTION: paral
#####
paral_kgb 0
#####
####                      SECTION: rlx
#####
optcell 0
```

Code for the P₂O₅ structure relaxation experiment

```
import os, shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import abipy.abilab as abilab
import abipy.flowtk as flowtk
from ase import Atoms
from ase.optimize import BFGS
from upet.calculator import UPETCalculator

from abipy_functions import gs_input, build_flow
# gs_input describes an ABINIT input script in Python format
# build_flow is an utility function managing flowtk.Flow used in AbiPy

# base_dir/def_dir setup
Si_file = os.path.join(def_dir, f"P2O5.cif") # Materials Project structure file
si_dir  # directory for storage of perturbed P2O5 cells
workdir # ABINIT working directory
traj_dir # Output of ASE trajectory files
upet_dir # Storage of UPET-optimised perturbed structures
ps_dir  # ABINIT pseudopotential files directory
# DIRECTORY MANAGEMENT
if (not si_dir): os.mkdir(si_dir)
if (not os.path.isdir(traj_dir)): os.mkdir(traj_dir)
if (not os.path.isdir(upet_dir)): os.mkdir(upet_dir)
if os.path.isdir(workdir): shutil.rmtree(workdir)

base = abilab.Structure.from_file(Si_file) # Structure load
structures_init = [] # perturbed, unoptimised structures
structures_upet = [] # perturbed structures optimised with UPET

count = 50    # number of initial perturbed structures
rerun = False # Whether we re-do perturbation and UPET optimisation
force_max = 0.005

# FIXED RANDOM NUMBER GENERATION - USED AS SEEDS FOR STRUCTURE PERTURBATION
```

```

generator = np.random.default_rng(42)
perturbation_seeds = generator.integers(1, 1000, size=count)

for i in range(count):
    perturbed_path = os.path.join(si_dir, f"si_{i}.cif")

    # Only generate perturbed structures if rerun is True OR none already exist
    if (os.path.exists(perturbed_path) and (not rerun)):
        structures_init.append(abilab.Structure.from_file(perturbed_path))
    else:
        # AbiPy method, distance in Angstrom
        structure = base.perturb(distance = 0.1, seed = perturbation_seeds[i])
        structures_init.append(structure.copy())

        # Write to file the perturbed structure
        path_to_file = os.path.join(si_dir, f"si_{i}.cif")
        structures_init[i].to(filename = path_to_file, fmt = "cif")

# UPET RELAXATION SETUP

oam = UPETCalculator(model="pet-oam-l", device="cpu")

for i in range(len(structures_init)):
    upet_relaxed_path = os.path.join(upet_dir, f"upet_si_{i}.cif")

    # Only run UPET optimisation if rerun is True OR it has not been run
    if (os.path.exists(upet_relaxed_path) and (not rerun)):
        structures_upet.append(abilab.Structure.from_file(upet_relaxed_path))
    else:
        structure = structures_init[i]
        s = structure.to_ase_atoms(calc=oam)
        traj_path = os.path.join(traj_dir, f"si_{i}.traj")

        # If an ASE optimisation stopped by error, we can re-use old trajectories
        if (os.path.exists(traj_path) and (not rerun)):
            dyn = BFGS(s)

```

```

        dyn.replay_trajectory(traj_path)
    else:
        dyn = BFGS(s, trajectory=traj_path)

    dyn.run(fmax = force_max) # ASE structural optimisation

    s.calc = None
    abi_struct = abilab.Structure.from_ase_atoms(s)
    abi_struct.to(filename = upet_relaxed_path, fmt = "cif")
    # The structure is detached from the calculator and written to file
    structures_upet.append(abi_struct)

# ABIPY/FLOWTK WORKFLOW

flow_inputs_1 = [
    gs_input(s, ecut=30, opt=0, bands=nband, pseudo_dir = ps_dir)
    for s in structures_init]
flow_inputs_2 = [
    gs_input(s, ecut=30, opt=0, bands=nband, pseudo_dir = ps_dir)
    for s in structures_upet]

flow = flowtk.Flow(workdir = workdir)
relax_work1 = [
    flowtk.RelaxTask(flow_inputs_1[i])
    for i in range(len(flow_inputs_1))
]
for i in relax_work1: flow.register_relax_task(i)
print("Structures_init loaded")

relax_work2 = relax_work = [
    flowtk.RelaxTask(flow_inputs_2[i])
    for i in range(len(flow_inputs_2))
]
for i in relax_work2: flow.register_relax_task(i)
print("Structures_upet loaded")

flow.make_scheduler().start()

```

