Skoltech
Skolkovo Institute of Science and Technology

MACHINE-LEARNING INTERATOMIC POTENTIALS FOR MULTICOMPONENT ALLOYS

Doctoral Thesis

by

KONSTANTIN GUBAEV

DOCTORAL PROGRAM IN MATERIALS SCIENCE AND ENGINEERING

Supervisor
Professor Alexander Shapeev

Moscow - 2019

Machine-Learning Interatomic Potentials for Multicomponent Alloys

Konstantin Gubaev
Skolkovo Institute of Science and Technology

Abstract

Atomistic simulations are a useful and sometimes irreplaceable tool which helps in providing insights into the structure and behavior of materials, which are often hard or impossible to obtain with other methods. For sufficiently accurate simulations of this kind, quantum-mechanical (QM) models of interatomic interaction are used. However, the computational complexity of QM models is such that they can be used for simulations of up to few hundreds of atoms at most.

This thesis describes the atomistic modeling approach, which relies on using a special type of machine-learning interatomic potentials—the moment tensor potentials (MTPs)—for approximating the quantum-mechanical potential energy surface using non-linear functions of local descriptors of atomic environments. In conjunction with the active learning algorithm of optimal training set construction this allows for performing simulations (molecular dynamics or structure relaxations) with much faster evaluation of energies, forces, and stresses in atomistic systems, referring to costly QM calculations only for the training data generation.

The proposed approach is applied to the search for the stable Cu-Pd, Co-Nb-V, and Al-Ni-Ti alloys, and to the prediction of the properties of small organic molecules. These results show high approximation capabilities of the MTPs and significant speed-up in searches for the new structures in comparison with approaches relying on DFT only.

Publications

1. K. Gubaev, E. V. Podryabinkin, G. L. Hart, and A. V. Shapeev. Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Computational Materials Science*, 156:148-156, 2019.
2.K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev. Machine learning of molecular properties: Locality and active learning. *The Journal of Chemical Physics*, 148(24):241727, 2018.

Acknowledgments

First of all, I want to thank my supervisor Alexander Shapeev for accepting me to Skoltech as his PhD student and for introducing me to the field of atomistic modeling. Without his guidance, especially at the early stages of my research, I would not have been able to achieve the results I present in my thesis. Also, I want to thank my colleagues from the Multiscale Modelling group at Skoltech: Ivan Novikov, Evgeny Podryabinkin, Evgenii Tsymbalov, Tatiana Kostiuchenko, Edgar Makarov, and Mozhdeh Shiranirad. I have learned something good from each of you, and I hope backward is true as well.

These acknowledgments would not be complete without mentioning the Skoltech professors, which made my (and not only mine) life in Skoltech interesting and productive. I want to thank Ivan Oseledets (together with Evgeny Frolov and Maksim Rakhuba) for the Numerical Linear Algebra course, which is exemplary brilliant and, in my opinion, is absolutely necessary for any tech-related educational institution. Also, I would like to thank Zeljko Tekic, an ultimately inspiring man and teacher, who continuously puts his effort into making innovators out of scientists—not many people can (and do) that. I give my special thanks to Artem Oganov, a prominent researcher and a wonderful pedagogue: every lecture he delivers is an immersion into a miraculous and astonishing world of science.

Contents

List of abbreviations

QM - quantum-mechanics
DFT - density functional theory
MD - molecular dynamics
MC - Monte Carlo
MLIP - machine-learning interatomic potential
EAM - embedded atom method
GGA - generalized gradient approximation
MAE - mean average error
RMSE - root-mean square error
BFGS - Broyden – Fletcher – Goldfarb – Shanno
PES - potential energy surface
CSP - crystal structure prediction
AL - active learning
HOMO - highest occupied molecular orbital
LUMO - lowest unoccupied molecular orbital

List of the machine-learning models of interatomic interaction

MTP - moment tensor potential
MPNN - message passing neural networks
DTNN - deep tensor neural networks
HIP-NN - Hierarchically Interacting Particle Neural Network
GAP - Gaussian approximation potential
SOAP - smooth overlap of atomic positions
HDAD - histogram of distances, angles and dihedrals
BOB - bag of bonds
BAML - bonds, angles, machine learning

MBTR - many-body tensor representations

MTM - moment tensor model

nlMTM - non-local moment tensor model

List of Figures

List of Tables

1 Introduction

Over the last decades it has been a growing demand for the new materials possessing some unique properties required by arising complexity of technological solutions. Designing a spacecraft, an artificial organ or more powerful electronic device are critical yet challenging goals requiring a great mobilization of developers from science and engineers from industry. Moreover, such technological solutions cannot be implemented without a great diversity of materials with specially hand-picked properties, such as metallic alloys. Designing the concept of a new material, creating its physical samples and testing them requires a lot of efforts. The traditional materials development is guided though trial-and-error approach, varying species concentrations and/or other conditions and measuring the properties of the obtained material, which results in a big amount of intermediate attempts and non-guaranteed results. Moreover, due to big amount of constituting components in contemporary alloys, the number of potential species concentrations (and, hence, resulting alloys) is enormous, and taking into account a relatively slow throughput of testing facilities, the development cycle of a new multicomponent alloy can constitute years or decades. Thus, the necessity for more thoughtful and selective strategy in alloys design is arising.

Recent advances in computer hardware and development of quantum-mechanical (QM) packages, allowing for accurate calculation of materials properties, created an opportunity to replace certain amount of laboratory work by simulations, thus allowing to speed-up the search for new materials. Depending on the purpose of the study and the phenomena to be investigated, these simulations involve different time and length scales, and, respectively, different modeling methods. To simulate the properties a bulk material at the engineering scale (e.g, a girder or a bridge) the continuum models are used, e.g., finite elements or finite volume models. They are based on solving governing differential/integral equations for continuum media on space and time grids. A vast of physical processes including heat, mass, and charge transfer or behaviour of material under the load are studied with continuum models.

Continuum models require a prior knowledge about the materials properties, e.g., transport coefficients, and are suitable for simulating properties of the constructions from already known materials.

Assessing the properties of a new material needs to be done by simulating processes occurring on the atomic scales (about 1 nm or less). If one needs to calculate a phase transition temperature, elastic constant or a phonon spectrum of some material, he or she needs to treat the material as a system of interacting particles, either atoms or molecules. In such case, one of the most common choices is to perform a molecular dynamics (MD) simulation, solving the Newton's equations for each particle at each time step. For this purposes a model of interatomic interaction is required, which would predict forces acting on atoms, depending on the atomic positions.

Models of interatomic interaction fall into two categories. The first category is quantum-mechanical models, often called *ab initio* or first-principles models. They allow calculations of high accuracy and are fully transferable, i.e., can in principle be applied to study of any atomistic system. Disadvantage of this type of models is their slow computational speed: even with the most powerful computational facilities, performing a QM simulation of the system containing more than a thousand of atoms over more than a nanosecond of physical time is infeasible. This way, only small length and time scales can be approached with QM models. The second type of models describing the interatomic interaction is empirical (or semi-empirical) models. As opposite to QM models, empirical models allow fast calculations of large atomistic systems (up to millions of atoms) over the long times (up to milliseconds), but their accuracy is often not sufficient for a quantitative analysis. In addition, empirical models by their nature do not take into account the states of the electrons, thus some electron-related properties like band gap, magnetic or optical properties usually can be investigated only with QM models. More detailed overview of interatomic interaction models is provided in Section 2. Molecular dynamics and Monte Carlo (MC) simulations using either QM or empirical models are successfully applied to solving of many problems in materials science. At the same time, a lot of processes like phase transitions or dislocations movement require accuracy of QM models and computational speed of empirical models to be properly simulated. Numerous attempts to design approximate QM models were made, but none of them resulted in a model with favourable accuracy/speed trade-off.

There is a bunch of atomistic simulation problems for which both QM and empirical models are unsuitable, which leaves the space for a new class of interatomic interaction models: the machine-learning interatomic potentials (MLIPs). The idea of MLIP is to construct a parameterized functional form describing a dependence of the potential energy (or any other property of interest) of the atomistic system on the positions and types of the constituting atoms. This functional form possesses certain flexibility: by adjusting the values of its parameters it can approximate different relations. In the most typical case the potential energy of the atomistic system depending on the positions and types of constituting atoms is to be approximated. Once this approximation is constructed, the MLIP is said to be *trained* to reproduce QM (and sometimes experimental) data. Once the MLIP is trained, it is able to reproduce the reference data with much greater speed if compared to QM calculations. Despite the accuracy of such a MLIP would be essentially limited by the accuracy of reference data used, still in many cases it is much more better than of empirical models. The field of MLIPs is in active development now, and many research groups are pursuing this direction, as MLIPs can significantly extend the opportunities of MD/MC approaches in materials modeling. More information about usage of MLIPs for atomistic simulations can be found in Section 2.4.

The present thesis describes my research devoted to the development of special type of MLIPs and using it to solve the problems of computational materials science. The MLIPs described in this thesis were originally proposed for single-component systems by my supervisor Alexander Shapeev [76] and were extended by me to the case of multiple components. The special functional form of these MLIPs allows to systematically improve the accuracy in cost of increasing the computational time consumption. More details about the MLIPs used in my research can be found in Section 3. Essential part of the methodology used in my research is the *active learning* algorithm of the training set construction. It was originally proposed in [60] and was adapted by me to multicomponent systems. It allows to compose the training set in a way, which prevents MLIP from extrapolation, and can be used to sample the configurational space while simulation “on the fly” (see Section 3.6.3). The MLIPs developed in my research were applied to the discovery of Cu-Pd, Co-Nb-V, and Al-Ni-Ti stable alloys (Sections 4.1.1-4.1.3) and to the predictions of the properties of 5-component organic molecules (Section 4.2). The MLIPs described in this thesis, together with the active learning algorithm, are implemented in a C++ and Python software packages. The overview and basic functionality of the software are provided in Section 5. The concluding remarks are given in Section 7.

2 Review of the Models of Interatomic Interaction

In this Section an overview of different interatomic interaction models will be given. The term *interatomic interaction model* is sometimes replaced by the *force field* or *interatomic potential* terms. Note, that not every interatomic interaction model can be called interatomic potential, as it assumes the explicit functional form of the potential energy; instead, the interatomic interaction model can mean some procedure (see, e.g, Section 2.3), which results in a certain potential energy value for each atomistic system. Nevertheless, I still will use the term interatomic potential, implying the broader sense of it, not limited to a case of explicit analytic function.

Interatomic potential provides a functional dependence of the energy of the atomistic system (further I will call it *configuration*) on the positions and types of the constituting atoms:

$$E = E(x). \tag{1}$$

From here x will be used to denote the configuration, meaning $x = \{\boldsymbol{r}_i, z_i\}$, where $\boldsymbol{r}_i$ and z_i denote the position vector and the atomic number of the ith atom in configuration.

2.1 Empirical Potentials

The empirical (sometimes called semi-empirical) potentials are used to avoid solving the governing quantum-mechanical equations. Instead, they provide an explicit analytical form with some coefficients found from fitting experimental or quantum-mechanical data. Majority of the empirical potentials are *local*, i.e., they account for interaction of each atom only with its nearest neighborhood, usually within some cutoff radius. The non-local potentials are usually used for charged systems, where the contribution of the long range Coulomb forces is essential. For local potentials the energy E is partitioned into the contributions V of individual atomic neighborhoods. Function V is called an *interatomic potential.* To define a neighborhood of the ith atom $\mathfrak{n}_i$, we let $\boldsymbol{r}_{ij}$ be the position of jth atom relative to the ith atom (thus, $\boldsymbol{r}_{ij}$ is a vectorial quantity) and z_j be the type of the jth atom. Then the neighborhood of the ith atom $\mathfrak{n}_i$ is the collection of $\boldsymbol{r}_{ij}$ and z_j, and the (1) can be rewritten as:

$$E(x) = \sum_i V(\mathfrak{n}_i), \tag{2}$$

The locality of the potential is expressed by the requirement that $V(\mathfrak{n}_i)$ depends only on atoms that are closer to the atom i than some cutoff distance R_{cut}, which is usually around 5 Å. In metallic systems this typically means the closest neighbors up to some (second, third, or even fourth) order are included in the atomic environment. An illustration of an atomic neighborhood is sketched in Figure 1.

The differences between various local potentials are contained solely in the function V. Below some of the most widespread potentials are listed:

1. Pair potentials

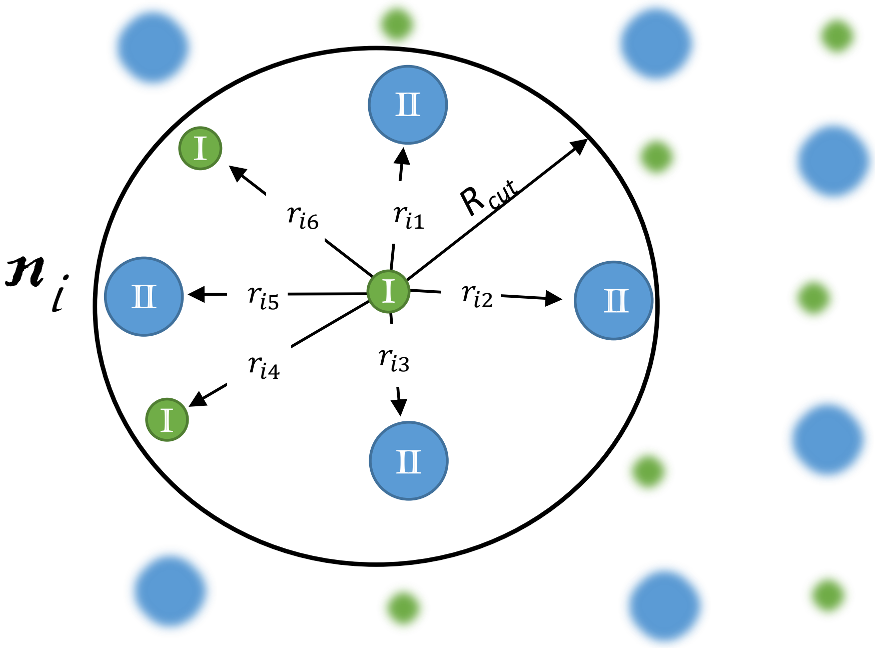


Figure 1: Partitioning scheme: energy E is composed from the contributions V_i of neighborhoods $\mathfrak{n}_i$. The neighborhood $\mathfrak{n}_i$ of the ith atom is described by the relative positions $\boldsymbol{r}_{ij}$ and the types z_j (I or II in this illustration) of the neighboring atoms.

Pair potentials by their nature include only two-body interactions, i.e., the total energy of a configuration can be written as the sum of the all two-body contributions:

$$E = \sum_i \sum_{j>i} \phi(r_i, r_j), \tag{3}$$

where condition $j > i$ is introduced not to consider each pair of interacting particles twice. Alternatively, (3) can be written in the local form:

$$V(\mathfrak{n}_i) = \frac{1}{2} \sum_{j \neq i} U(r_{ij}), \tag{4}$$

where function $U(r_{ij})$ is called a *pairwise potential.* Some of the pair potentials are:

Lennard-Jones potential:

$$U(r_{ij}) = 4\epsilon((\sigma/r_{ij})^{12} - (\sigma/r_{ij})^6). \tag{5}$$

Morse potential:

$$U(r_{ij}) = \epsilon(e^{-2\alpha(r_{ij}-r_0)} - 2e^{-\alpha(r_{ij}-r_0)}) \tag{6}$$

Note, that Lennard-Jones and Morse potentials can be used as non-local potentials as well. However, typically these potentials are used with some cutoff radius to account for screening of interatomic interactions and therefore can be considered local.

2. Three-body potentials

   Three-body potentials account for three-body interactions:

$$E = \sum_i \sum_{j \neq i} U(r_i, r_j) + \sum_i \sum_{j \neq i} \sum_{k \neq i,j} W(r_i, r_j, r_k). \tag{7}$$

   These potentials include Tersoff and Stillinger-Weber potentials. While two-body terms sometimes are associated with bond lengths, three-body terms are usually associated with angles between bonds.

3. Many-body potentials

   The most widespread empirical potential for metals is the embedded atom method (EAM) potential [18]. It originates from the idea of embedding each atom into the electronic gas with a density, affected by other atoms. This interatomic potential has the following form:

$$V(\mathfrak{n}_i) = \frac{1}{2}\sum_{j\neq i}\phi(r_{ij}) + F(\rho_i), \tag{8}$$

   where $\phi(r_{ij})$ is the pairwise repulsive term, F is the non-linear (e.g., a polynomial) *embedding* function and ρ_i stands for the electron density at the position of the atom i and is composed from the contributions of the neighboring atoms:

$$\rho_i = \sum_{j\neq i} f(r_{ij}), \tag{9}$$

   where $f(r_{ij})$ is some scalar function. There are other many-body potentials resembling the form of EAM, e.g., Finnis-Sinclair potentials [80].

2.2 Quantum-Mechanical Models

The quantum-mechanical models are based on the solutions of different approximations of the time-independent Schroedinger equation:

$$\hat{H}\Psi = E\Psi. \tag{10}$$

Ψ in (10) means the total wave function of the system, including both nuclear and electronic degrees of freedom $\Psi = \Psi(r_i, R_i)$, where r_i and R_i are electrons and nuclei positions respectively. As in general case the solution of the equation (10) is not possible, the Born-Oppenheimer approximation is often used. It "freezes" the nuclei, as they move much slower than the electrons due to three orders of magnitude mass difference and comparable electrostatic forces. This approximation allows separating of the total wave function Ψ onto electronic $\psi(r_i)$ and nuclear $\chi(R_i)$:

$$\Psi(r_i, R_i) = \psi(r_i, R_i)\chi(R_i),$$

splitting the equation (10) onto electronic and nuclear parts which can be solved consequently.

Hamiltonian H of the system with N_e electrons interacting with each other ($U(\boldsymbol{r}_i, \boldsymbol{r}_j)$) and with an external field generated by the nuclei (V_i) can be written as:

$$\hat{H}\psi = \Big(\sum_{i=1}^{N_e}(-\frac{\hbar^2}{2m_e}{\nabla_i}^2) + \sum_{i=1}^{N_e} V(\boldsymbol{r}_i) + \sum_{i=1}^{N_e}\sum_{j>i}^{N_e} U(\boldsymbol{r}_i, \boldsymbol{r}_j)\Big)\psi = E\psi. \qquad (11)$$

The Born-Oppenheimer approximation greatly simplifies (10), as the resulting equation (11) should now be solved with respect to the electronic wave functions only. Still, (11) cannot be solved in general case and requires further simplifications, e.g., density functional theory (see Section 2.3). Though no electronic (or nuclear) spins were considered in the derivation of (11), they can be included in (11) directly as its derivation takes into account only spacial degrees of freedom.

There are approaches which search the solution of (11) in a form of a linear combination of atomic orbitals, e.g., Hartree-Fock, post-Hartree-Fock methods. Among these are tight-binding model which treats electrons as "belonging" to atoms and uses linear combination of atomic orbitals to determine the electron energy levels. The opposite (in a sense of binding to atoms) approximation - nearly free electron model (modification of a free electron model) treats electrons as a gas with only weak interaction with ions, which allows for correct prediction of many features of the electronic structure, especially in metals, when outer electrons are essentially delocalized. Hybrid functionals in DFT (Section 2.3) also employ predictions by Hartree-Fock theory, and they are capable of calculating many chemical systems with high accuracy, introducing corrections to the exchange-correlation functionals (Section 2.3). Apart from relatively "heavy" methods like DFT hybrid functionals or, etc., quantum Monte Carlo (which operates with exact many-body wavefunction and treats quantum effects directly) some QM models are semi-empirical and adjusted for relatively fast calculations, providing approximate yet often accurate predictions for large systems, which evaluation with more costly methods would be prohibitively long; e.g., linear scaling DFT, which incorporates some screening of interatomic interaction leading to $O(N)$ cost scaling with the number of atoms, versus $O(N^3)$ for conventional DFT.

2.3 Density Functional Theory

One of the most widespread approaches if density functional theory (DFT), which in particular is very useful for metals and alloys, providing an appealing trade-off between accuracy of calculations and computational speed. This method I used in my research to calculate *ab initio* properties of alloys, and due to aforementioned it deserves a special attention and more detailed description. Conventional DFT cost scales as $O(N^3)$ with the number of atoms, while it exists also a “linear scaling DFT” implementation with $O(N)$ scaling, less accurate but still applicable, especially for large systems.

The DFT is a QM approach based on the two Hohenberg-Kohn theorems:

1. The external potential (and hence the total energy) is a unique functional of the electron density.

2. The energy functional that delivers the ground state energy of the system, gives the lowest energy if and only if the input density is the true ground state density. For any positive integer N and potential $\nu(\boldsymbol{r})$, a functional $F[(\boldsymbol{r})]$ exists such that

$$E_{(\nu,N)}[n(\boldsymbol{r})] = F[n(\boldsymbol{r})] + \int \nu(\boldsymbol{r})n(\boldsymbol{r})d^3r$$

obtains its minimal value at the ground-state density of N electrons in the potential $\nu(\boldsymbol{r})$. The minimal value of $E_{(\nu,N)}[n]$ is then the ground state energy of this system.

Hohenberg-Kohn theorems allow to reformulate the (11) for a system of N_a atoms (nuclei) with charges Z_j and N_e electrons in terms of the electron density function instead of the wave function $\Psi(\boldsymbol{r}_1, ..., \boldsymbol{r}_{N_e}, \boldsymbol{R}_1, ..., \boldsymbol{R}_{N_a})$:

$$
\begin{aligned}
E[n(\boldsymbol{r})] =& T[n(\boldsymbol{r})] + \sum_{j=1}^{N_{atoms}} \int_{\mathbb{R}^3} \frac{e^2 Z_j n(\boldsymbol{r})}{|\boldsymbol{r} - \boldsymbol{R}_j|} d^3(\boldsymbol{r}) + \iint_{\mathbb{R}^6} \frac{e^2 n(\boldsymbol{r}) n(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} d^3(\boldsymbol{r}) d^3(\boldsymbol{r}') \\
&+ E_{XC}[n(\boldsymbol{r})] + \int_{\mathbb{R}^3} V_{ext}(\boldsymbol{r}) n(\boldsymbol{r}) d^3(\boldsymbol{r}),
\end{aligned}
\tag{12}
$$

where $T[n(\boldsymbol{r})]$ is the kinetic energy term, single and double integrals are the electron-nuclei and electron-electron Coulomb repulsion respectively, $V_{ext}(\boldsymbol{r})$ is the external potential, and term $E_{XC}[n(\boldsymbol{r})]$ includes other not precisely known factors such as exchange-correlation interaction.

The electronic density $n(\boldsymbol{r})$ is given by:

$$
n(\boldsymbol{r}) = \sum_{i=1}^{N_e} |\psi_i|^2.
$$

To account for spin degrees of freedom, similarly to (2.3) spin-up and spin-down densities can be introduced, calculated from corresponding spin-up and spin-down wave functions ψ_i.

Since the exact form of the term $E_{XC}[n(\boldsymbol{r})]$ is unknown, we can rewrite the equation (12) describing N_e interacting particles in some external potential $V_{ext}(\boldsymbol{r})$, as a a system of equations (Kohn-Sham equations, [88]) of non-interacting particles in the effective Kohn-Sham potential:

$$
\Big(- \frac{\hbar^2}{2m_e} {\nabla_i}^2 + V_{KS}(\boldsymbol{r}) \Big) \psi_i(\boldsymbol{r}) = e_i \psi_i(\boldsymbol{r}), i \leq N_e, \tag{13}
$$

where the Kohn-Sham potential includes the functionals, corresponding to the external potential $V_{ext}(\boldsymbol{r})$, the Coulomb interaction of electrons $\int \frac{e^2 n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d^3 r'$ and the exchange-correlation interaction between electrons $V_{XC}[n(\boldsymbol{r})]$.

$$
V_{KS}(\boldsymbol{r}) = V_{ext}(\boldsymbol{r}) + \int_{\mathbb{R}^3} \frac{e^2 n(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} d^3 r' + V_{XC}[n(\boldsymbol{r})] \tag{14}
$$

The form of $V_{XC}[n(\boldsymbol{r})]$ is still unknown, and all the effects not explicitly included in (14) are put into $V_{XC}[n(\boldsymbol{r})]$.

This way, the many-particle problem (11) is replaced by a set of single-particle problems (13) without any loss of generality. To solve (13) we need to find the wave functions ψ_i, assuming that we know $V_{KS}(\boldsymbol{r})$. $V_{KS}(\boldsymbol{r})$, in turn, depends on the electronic density distribution $n(\boldsymbol{r})$, which is uniquely defined by a set of ψ_i. Thus, the equations (13) need to be solved self-consistently, e.g., iteratively. Usually one starts from some initial guess for $n(\boldsymbol{r})$, then calculates the $V_{KS}(\boldsymbol{r})$ and solves the (13) with respect to ψ_i, which delivers the new density distribution $n(\boldsymbol{r})$. These steps are then repeated until convergence is reached.

Having the form of $V_{XC}[n(\boldsymbol{r})]$ unknown, different approximations are introduced. The simplest and the most computationally efficient is the local-density approximation (LDA), which assumes that $V_{XC}(r)$ depends only on $n(\boldsymbol{r})$ at this point of space, i.e., is local: $V_{XC}(\boldsymbol{r}) = V_{XC}(n(\boldsymbol{r}))$. The generalized gradient approximation (GGA) is more computationally demanding, but in many cases much more accurate. It takes into account the dependence on the spatial derivative of the electron density in a certain point: $V_{XC}(\boldsymbol{r}) = V_{XC}(n(\boldsymbol{r}), \nabla n(\boldsymbol{r}))$.

2.4 Machine-Learning Potentials

Machine-learning interatomic potentials (MLIPs) are a class of interatomic interaction models, which have large amount of numerical parameters, which are to be found from the training procedure (see Section 3.1). Note, that empirical potentials may also have adjustable parameters, e.g., ϵ and σ from (5) or coefficients of the expansion of embedding function from (8) or electron density function (9). Still, the MLIPs and empirical potentials differ in a number of ways:

1. Functional form

   Empirical potentials have functional forms which are physically inspired and reflect some analytical knowledge about the system, e.g., the energy of the system can be explicitly decomposed onto contributions of different physical factors, like presence of different types of bonds. MLIPs instead have functional forms motivated by usage of different types of regressors (kernel-based models with different kernels, neural networks).

2. Interpretation and number of parameters

   Empirical potentials typically have from few to few dozens of parameters, which often carry direct physical meaning, like the depth of the potential well ϵ and equilibrium interatomic distance σ in (5). As opposite, MLIPs involve hundreds, thousands or even tens of thousands of parameters which carry no direct physical meaning and are essentially some coefficients in the representation of the approximated function.

3. Universality

   Empirical potentials are designed for each system specifically, and adjusting of their parameters is done with a thorough control from the human side. Often a few versions of the same empirical potential for a certain system can be introduced, aimed at better reproduction of different physical quantities. MLIPs are much more universal in a sense that the procedure of finding their parameters is general and not case-dependent (see Section 3.1).

All the aforementioned differences are the natural outcomes of the idea of MLIPs: to provide a flexible functional form, which is able to accurately approximate the reference quantum-mechanical data within the training domain, and does this much faster than the QM calculations. The MLIPs available in the literature can be classified as the ones based on neural networks and the ones based on different kernels.

The neural network-based models include: MPNN (message passing neural networks [29]), DTNN (deep tensor neural networks [73]), HIP-NN (Hierarchically Interacting Particle Neural Network [50]) and SchNet [71]. The common feature of neural network-based models is large amount of internal parameters, which results in large datasets, required for a proper training (see

Section 2). At the same time, such models usually are capable of achieving high accuracy.

Some of the existing approaches are based on different kernels: kernels based on Coulomb matrix [68, 67], GAP (Gaussian approximation potentials [81], relying on Gaussian kernels), the SOAP (smooth overlap of atomic positions) kernel [19], HDAD (histogram of distances, angles and dihedrals [24]), BOB (bag of bonds [33]), BAML (bonds, angles, machine learning [39]), MBTR (many-body tensor representations [41]).

Even with all variety of ML methods developed for atomistic modeling in recent years, this only reflects the immaturity of the present MLIPs field and continuous attempts to search for more generic, more accurate and more robust approaches. Of course, all the ML methods enlisted in this section suit for certain problems they were developed for, however they possess certain drawbacks, motivating the development of yet another ML approach, e.g.: the GAP [81] requires much more computational time to achieve the same accuracy compared to the single-component MTP (see [76]), NN-based models usually require a lot of training data (see [32]), cluster expansion is limited to lattice-based structures. From my personal point of view, the drawbacks of contemporary MLIPs are a consequence of one of the following reasons: the concept of a MLIP can be too "physically-motivated" (e.g, EAM potentials) or vice-versa, largely inspired by ML and thus not well appropriate for describing the physical (atomistic) systems. Physically inspired models can suffer from lack of flexibility in their functional forms, thus failing to capture complex (features of interatomic interactions not reflected in "physics" of their design. On the other hand, ML-based models can neglect or not fully account for the origin of the data they are fit to, which results in excessive attempts required to capture basic physics phenomena (e.g., non-differentiable ML models like random forest predicting energy but not forces, which are essentially the derivatives of the energy). From this perspective the MTPs incorporate a balance between physically inspired descriptors and flexible polynomial functional form (see Section 3.4) making approximation of PES accurate with relatively small number of parameters [32] and allowing for making more accurate MTPs by employing more complex functional form. In addition, the active learning approach (Section 3.6) developed for MTPs solves the sampling problem (Section 3.2), which arises in any simulation involving ML models and can be more difficult to overcome, then the problem of accurate MLIP fitting.

2.5 Discussion

Models of interatomic interaction are designed to perform simulations of material properties on the atomic level. The simulations can vary in their space and time scales, number of particles, types of the atomistic systems, desired accuracy, etc. Depending on the concrete simulation scenario, different types of interatomic interaction models can be used. For accurate simulation of small (up to hundreds of atoms) atomistic systems the quantum-mechanical models (among which DFT is very popular) are used. For the simulations involving large time scales and large numbers of atoms (up to millions) the only choice is using empirical potentials, which in general provide moderate accuracy, sufficient for qualitative analysis only at best.

Contemporary technological challenges are connected with materials properties, which often cannot be properly simulated either by empirical or quantum-mechanical potentials, as they require conducting numerical experiments of both high accuracy and large numbers of atoms (e.g., nucleation events or dislocation movements). At the same time, machine-learning interatomic potentials exploit hidden relations in quantum-mechanical data, which allow reproducing of computationally expensive quantum-mechanical calculations at comparatively low cost. MLIPs therefore are a promising alternative to QM and empirical models, as they potentially enable simulations of quantum-mechanical accuracy and computational speed comparable to that of empirical potentials.

3 Methodology of Machine-Learning Potentials

In this Section I provide in detail the mathematical background of machine-learning interatomic potentials (MLIPs), the main problems connected to the usage of MLIPs and the ways of solving them. As well, in this Section the moment tensor potentials (a special type of MLIPs I use in my work) are presented, together with the active learning approach to training. In this section by "we" I mean me and my colleagues (most likely my supervisor), with whom I developed this methodology.

3.1 The Training Procedure

We first formulate a problem of fitting a MLIP on some dataset in a machine learning framework. Suppose there is a large number, N, of configurations whose structure (and composition) is encoded by $x^{(1)}, \ldots, x^{(N)}$. The task is to construct a function F that predicts a certain property (e.g., an atomization energy) of each configuration as $F(x^{(1)}), \ldots, F(x^{(N)})$. The function F, which I refer to as the *model*, is constructed based on data—the properties of the first n ($n < N$) configurations, $y^{(1)}, \ldots, y^{(n)}$, which are called the *labels.* We call the set $\{x^{(1)}, \ldots, x^{(n)}\}$ together with $\{y^{(1)}, \ldots, y^{(n)}\}$ the *labeled dataset* or the *training set.* The labeled dataset is often chosen randomly. The model has a number m of free parameters $\boldsymbol{\theta} = (\theta_1, .., \theta_m)$. We organize them in a vector $\boldsymbol{\theta}$ of length m, which is found from minimizing the total loss functional

$$L(\boldsymbol{\theta}) = \sum_{j=1}^{n} \big(y_j - F\big(\boldsymbol{\theta}, x^{(j)}\big)\big)^2 \longrightarrow \min \tag{15}$$

in a procedure called *training* (see Section 3.1). Once the solution $\bar{\boldsymbol{\theta}}$ of the problem (15) has been found, the accuracy check is performed, typically via calculating the so-called *mean average error (MAE)* or *root-mean square error (RMSE)* on some set of configurations $\{X^{(1)}, \ldots, X^{(K)}\}$ with known properties $\{Y^{(1)}, \ldots, Y^{(K)}\}$:

$$\mathrm{MAE} = \frac{1}{K}\sum_{j=1}^{K} |Y_j - F\big(\boldsymbol{\theta}, X^{(j)}\big)|, \tag{16}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{K}\sum_{j=1}^{K} \big(Y_j - F\big(\boldsymbol{\theta}, X^{(j)}\big)\big)^2}. \tag{17}$$

If the errors (16) are measured on the training set, they are called *training errors* and reflect the quality of fitting.

The key components of training and the possible problems which should be taken into account are described below.

1. Functional form

   This is the factor of the model $F(\boldsymbol{\theta})$ itself, namely its functional form, meaning characteristics of the algebraic functions incorporated in $F(\boldsymbol{\theta})$.

   <u>Unsuitable representation</u>

   If you approximate a function $y(x) = x^8 + x^4$ on the interval $(-100, 100)$ with odd polynomials $F(\boldsymbol{\theta}, x) = \theta_1 x^5 + \theta_2 x^3 + \theta_3 x$ you will completely fail, as $y(x)$ is an even function. The solution is to change the representation of $F(x)$. Another problematic case is fitting a discontinuous function, like a Heaviside function $H(x)$, with continuous model $F(x)$.

   <u>Insufficient amount of parameters</u>

   If you approximate a function $y(x) = x^8 + x^4$ on the interval $(-100, 100)$ with polynomials of order 5: $F(\boldsymbol{\theta}, x) = \theta_1 x^5 + \theta_2 x^4 + \theta_3 x^3 + \theta_4 x^2 + \theta_5 x + \theta_6$ you will also fail, as $F(\boldsymbol{\theta}, x)$ is a lower order polynomial compared to $y(x)$, thus an accurate fit is not possible. The solution is to include polynomials of higher order into the representation of $F(x)$. The problems of this kind are sometimes called an *underfitting* (see Figure 4a).

2. Training set

   This factor includes peculiarities of the approximated data. In practice, $y(x)$ is obtained via some procedure (either experimental or computational), with a possibility of:

Inconsistency of data

This problem can arise in a case of mixing data from different sources, e.g., experimental and computational data, or even computational data obtained with different methods. If for some close values x_0 and x_1 the corresponding values of $y(x_0)$ and $y(x_1)$ differ much, it can result in bad convergence of the optimization algorithm and, moreover, unphysical behaviour of the trained model. Also, as energy is defined up to an arbitrary constant, these constants may vary for different computational models (e.g., different pseudopotentials in DFT). The problem of data inconsistency is illustrated in Figure 2: combining data from different sources leads to incorrect deformation energy dependence learned by MLIP.

Noisy data

By the term "noisy data" the data with uncertainty is meant, when for each argument x different values $y(x)$ can be observed (or measured): $y(x) = x^8 + x^4 \pm \epsilon(x)$, see Figure 3. The noise present in data can be of different kind and different techniques are used to cope with it. Some of the techniques are similar to the ones which prevent overfitting, e.g., regularization or *early stopping* techniques (see this section further).

Not properly sampled data

This means the data points present in the training set do not cover some vital parts of the input parameters region. E.g., if you approximate a function $y(x) = x^4 + x^3 + x^2 + x + 1$ at x points of this kind: {-500,-400,-200,200,300,600} you will only approximate the left and the right branches of the $y(x) = x^4$ curve, but not the up-down oscillation near the roots of the equation $x^4 + x^3 + x^2 + x + 1 = 0$, as much denser points concentration near $x = 0$ is needed to capture it. This problem is discussed in more details in Section 3.2 and is illustrated on Figure 5.

3. Optimization procedure

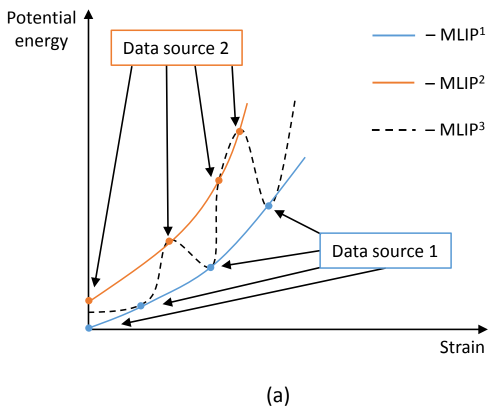
The algorithm of solving the optimization problem (15) can significantly affect the training errors. Most likely, the parameters of MLIP $\boldsymbol{\theta}$ can take continuous values ranging from $-\infty$ to $+\infty$. Thus, training of the model $F(\boldsymbol{\theta})$ with m parameters means finding the best parameters out of the space $\boldsymbol{R}^m$, which is not a trivial task.

Ideally, while minimizing the functional (15) one wants to find the best solution among the all possible ones, i.e., the *global minimum*. The problem is then called a global optimization problem. However, in general case it is unsolvable in principle, as even if there are 100 (instead of infinite number) different values for each component of $\boldsymbol{\theta}$, for $m = 100$ one should calculate 100^{100} values of the functional (15), which is not possible due to the limitations on computational resources. Exact solution o the optimization problem $L(\boldsymbol{\theta})$ can be found only in case of polylinear or polyquadratic $L(\boldsymbol{\theta})$. i.e., the minimizing functional has linear/quadratic dependence on the internal parameters. In general, the algorithm of finding a global minimum for problems like (15) with nonlinear and non-quadratic dependence of the $L(\boldsymbol{\theta})$ (15) w.r.t. model parameters does not exist. Therefore in practice the training procedure is aimed on finding the sufficiently good *local minimum*, out of potentially infinite number of all local minima. For this task algorithms of local optimization are used.

Majority of the local optimization algorithms require derivatives of the loss function w.r.t. model parameters $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ to solve the problem (15), as it provides a direction in a parameter space, along which the function (15) can be decreased. One of the algorithms which do not require the calculation of the derivatives $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is the Nelder-Mead [20] algorithm. In my research I use one of the most popular algorithms, the Broyden – Fletcher – Goldfarb – Shanno (BFGS) [36] algorithm, as it shows better convergence that the conjugate-gradient [79] or Nelder-Mead algorithms.

Any of the local optimization algorithms will converge to some local minimum with no guarantee of it being better than other local minima. Therefore, to solve a global optimization problem, some extra techniques are used to search for several possible solutions in different regions of parameter space $\boldsymbol{R}^m$. Unlike genetic algorithms [28], which solve the global optimization problem as it is, one can use the *simulated annealing* [85] or the *basin-hopping* [86] techniques to acquire different local minima with the local optimization algorithms. This means that for a fixed model $F(\boldsymbol{\theta})$ and a fixed training set $\{x^{(1)}, \ldots, x^{(n)}\}$ one can find several sets of parameters $\boldsymbol{\theta}$, each corresponding to some particular value of $L(\boldsymbol{\theta})$. The $L(\boldsymbol{\theta})$ in this case is a measure of accuracy: the smaller $L(\boldsymbol{\theta})$ is, the more precise approximation of the training set takes place.

Another difficulty which can arise in training of a ML model is the *overfitting* problem. It happens when a ML model, being trained on a particular training set, learns the artificial features of this set, not reflecting the actual relations behind the data, see Figure 4c. Typically this occurs when the number of parameters, m, is too large. While training errors measure the accuracy of approximation of the training set itself, the capabilities of MLIP to approximate the data, not contained in the training set should be measured on a separate set not used for training. Usually such set is called *holdout set* or *testing set*. Sometimes some *validation set* (not intersecting with either training set or holdout set) is used to control the accuracy during training, stopping the optimization procedure when the accuracy on the validation set starts to decrease, which is a sign of overfitting. This is the early stopping technique widely used for deep neural networks (which can have orders of $10^4 - 10^5$) parameters. Such terms as holdout set, validation set, and testing set can be used interchangeably in literature. If the errors (16) are measured on a set of configurations, not intersecting with the training set, in this thesis they are called *validation errors*.



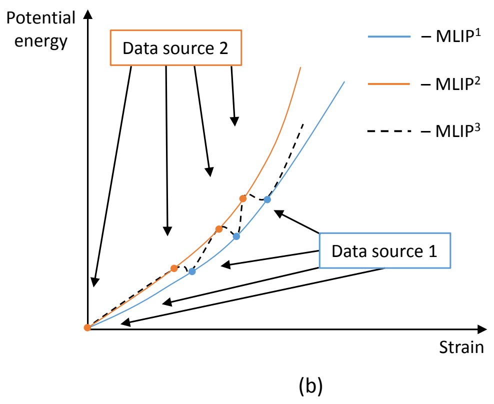


Figure 2: Illustration of the problem of fitting MLIP to the energies of strained configurations obtained from different sources of data. Both MLIP^1 and MLIP^2 fitted to the data from corresponding sources reproduce physically correct curves with deformation energy growing monotonically with increasing strain. MLIP^3 is fitted to the combined data, which results in unphysical dependence of the deformation energy on strain due to the inconsistency of data. On the graph (a) the energy values from different data sources have different energy shifts. On the graph (b) the energy shifts are equalized, but different forms of potential energy dependencies still result in an unphysical behaviour of MLIP^3.

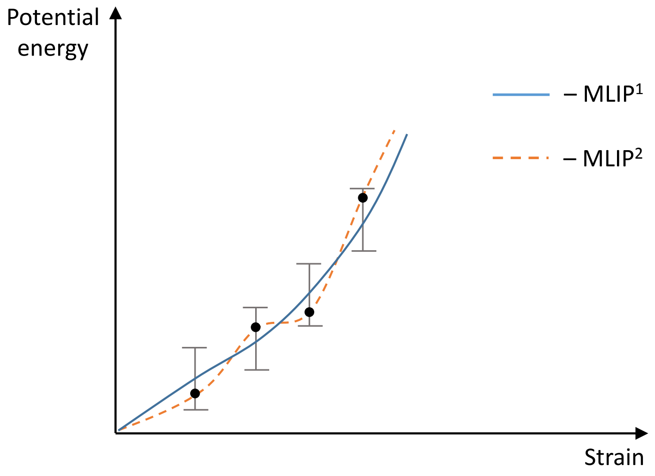


Figure 3: Illustration of the problem of fitting MLIP to the energies of strained configurations, calculated with noise. MLIP^2 suffers from overfitting, as it learns the non-existing features of the data, induced by the noise. MLIP^1 learns the noise-averaged data, which can be achieved by regularization or early stopping as well as by selecting a MLIP with less number of parameters.

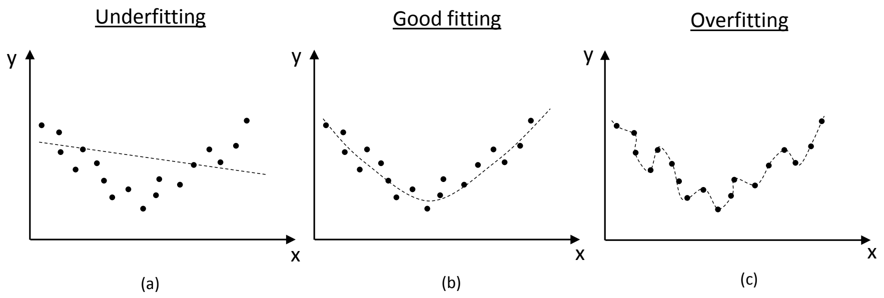


Figure 4: If model has too few parameters, underfitting (a) occurs, i.e., the model is not flexible enough to accurately represent the data. If model has too many parameters, overfitting (c) can occur, i.e., the model has too much flexibility and learns the specific features of the training set. The optimal fit (b) is therefore a matter of choosing the proper model and the proper training set.

3.2 Constructing a Machine-Learning Potential

Any MLIP, before it can be used to predict certain properties of atomistic systems (typically these are energies, forces and stresses) should first be trained, e.g., some values should be assigned to parameters $\boldsymbol{\theta}$. Then the MLIP is able to approximate the quantum-mechanical property F^{qm} of configuration x:

$$F(\boldsymbol{\theta}, x) \approx F^{qm}(x) \tag{18}$$

The multidimensional space consisting of configurations with all possible combinations of atomic positions and types (relevant to a certain simulation) is called the *configurational space*. The dependence $F(x)$ or (in the most frequent case when F means energy) $E(x)$, which means the mapping of each point of configurational space x to some specific value of energy $E(x)$, is called the *potential energy surface (PES)*. Solving the optimization problem (15) delivers the best set of parameters $\boldsymbol{\theta}$ (in some sense) to approximate the PES.

A typical way of using MLIPs is to employ them as force-fields in MD/MC simulations. The problem is that the set of configurations $\{x^{(1)}, \ldots, x^{(n)}\}$, which the MD will "visit", is not known in advance. As MLIPs allow much faster calculation of the properties $F(\boldsymbol{\theta}, x^{(j)})$ then QM packages do, MLIPs are typically used in simulations involving much larger system sizes and simulation times, than those attainable with QM packages. Thus, it is often not possible to systematically measure the accuracy of certain MLIP during the simulation, as generating QM data for all respective configurations is too computationally expensive and experimental data might not exist at all. Sometimes indirect criteria are introduced, like correspondence of some physical quantities calculated during the simulation to their reference values. However, when the simulation is aimed on providing new information, which is not available from other sources, the question of reliability of the results remains open, as there are no robust methods of checking the accuracy (and, therefore, applicability) of MLIP in a certain simulation. The same problem arises for the empirical potentials as well.

In a situation, when the direct measurement of MLIP accuracy is not possible, each user relies on its own experience to construct a reliable MLIP. By "construction" the following procedure is meant:

1. Preparation of the training set

   It is worth to note that MLIP, being essentially a machine-learning model, can only interpolate the relation $y_j = F^{qm}(x^{(j)})$ incorporated in the provided training set, but not more than that. In the Figure 5 you can notice three cases of approximating a PES of a system containing two atoms, acting via the Lennard-Jones potential (5). On the Figure 5a the part of configurational space, covered by the training set (the *training domain*) does not contain samples with small and big r, thus leading to incorrect behaviour of the trained MLIP at the corresponding distances. On the Fig.5b the training domain does not cover the potential energy well, and, consequently, the trained MLIP fails to reproduce it. On the Figure 5c the training samples are chosen in a proper way, thus MLIP learns all the features of the approximated PES. From this 1-dimensional example it may seem that a simple uniform coverage of the configurational space is the best choice to form a training set. But in a more realistic case of fitting a many-body local MLIP to the PES of a bulk metal using a cutoff radius of 5 $\mathring{A}$, the number of atoms in the neighborhood (see Figure 1) would be about few dozens, let us take 20. In a case of one component, each atom would have three scalar degrees of freedom, thus leading to about 60 arguments in a function $V(\mathfrak{n})$ (not exactly 60 due to the symmetries). If the minimum distance between two atoms equals 2 $\mathring{A}$, and the distances on the interval from 2 to 5 $\mathring{A}$ are discretized with 0.1 $\mathring{A}$ step, this leads to a 30 points per distance and to 30^{20} configurations required for the uniform sampling, which is by far impossible. This problem is called *a curse of dimensionality*, and it arises from a high amount of independent variables (about 100 in our case) on which the approximated function depends. As far as the uniform sampling is prohibited by a curse of dimensionality, typically other approaches are undertaken.

2. Sampling

   As illustrated by Figure 5, the MLIP will approximate PES well within the training domain (still one should care about the underfitting or overfitting problems, see Figure 4). Thus, while designing MLIP for simulations of particular atomistic systems, the training set should be composed from configurations resembling the ones expected to occur during the simulation. The procedure of collecting such configurations is called *sampling*. While sampling some force field is used to run an MD and collect the snapshots (the MD trajectory). Afterwards this some representative subset of snapshots is selected (namely their geometries) and put into *ab initio* calculations to obtain the reference energies, forces and (sometimes) stresses.

   For example, to simulate a crystalline structure of a bulk iron at 300K, a training set should consist of configurations with lattice type and lattice parameter close to the ones of iron at 300K (BCC lattice with a 2.856$\mathring{A}$ lattice constant). During simulation the atoms will displace from their equilibrium positions due to thermal fluctuations, and the proper training set should include not only on-lattice configurations, but the "shaken" ones as well. The sampling can be done with a Lennard-Jones potential with relevant parameters, by performing an MD simulation with NVT ensemble and saving the trajectory. Next, some amount of configurations is picked, corresponding to time steps not close to each other (in order to exclude correlated configurations), and after calculation of *ab initio* energies/forces/stresses these configurations enter the training set.

In this sampling approach it is assumed that even with some simple (therefore computationally fast) interatomic interaction model MD can "visit" appropriate configurations, which would provide necessary input for training of the model. Still, in many cases this is only the first step, while the next is to use the MLIP, trained on such training set, to run another MD, saving its trajectory. Then a new training set is obtained and a new MLIP is trained. This iterative procedure can be repeated several times, till the MLIP is fully trained; however, there is no strict and systematic criterion for this condition.

3. Validation of MLIP

   To check the MLIP capabilities of describing the atomistic systems of interest, typically some simulations are performed, aimed at computing some materials properties, which then can be compared to known experimental/theoretical data. Note, that a small error on the training set does not guarantee the good performance of MLIP, as it will be used to calculate the properties of configurations not present in the training set.

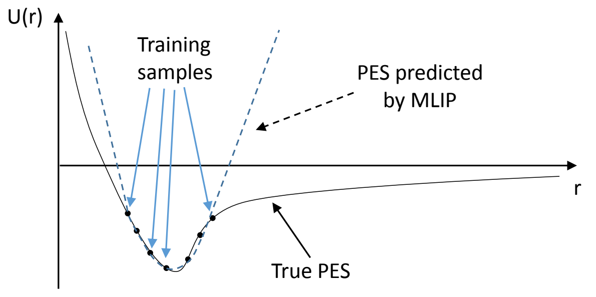
   A common way of checking the performance of the potential outside the training set is using a validation set (see Section 3.1). However, it is not always possible to construct a validation set. It requires configurations relevant to the simulation of interest with provided *ab initio* energies/forces/etc., which is not always possible. The reasons can be: the potential can be used for large supercell calculations, for which *ab initio* calculations are not possible, or the configurations from the simulation are not known in advance, because the trajectory of the MD is known only during actual simulation. Therefore, sometimes validation of the potential is done versus more "high-level" quantities like vacancy migration energy barrier, elastic moduli or heat capacity: the potential can be used to calculate some of these quantities with further comparison to their known values. This allows for estimation of the potential quality: if the potential reproduces some quantities well, it can be expected that it will properly predict other properties of interest. Obviously this is not a systematic criterion as opposite to the active learning approach, see Section 3.6.

3.3 Energies, Forces, and Stresses

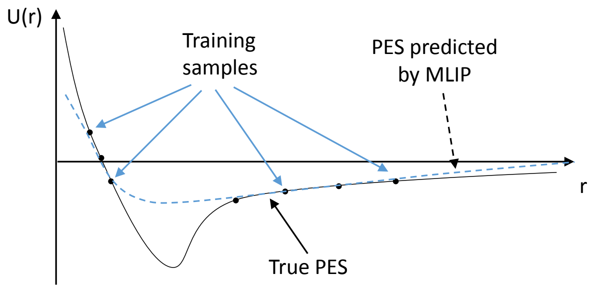
We next describe in detail the model used in the present thesis and its functional form F. In (15) the model F can approximate any property of the configuration. We consider the energy as an approximated property and thus we use E to denote it. Next, by the machine-learning model I mean the moment tensor potentials (MTPs) described in the Section 3.4. They are parametrized with a set of parameters $\boldsymbol{\theta}$ which are found from minimizing the loss functional, expressing that the predicted energy E is close to the reference quantum-mechanical energy E^{qm}. The loss functional (15) is then written as:

$$L(\boldsymbol{\theta}) = \sum_{j=1}^{N} \Big(E\big(\boldsymbol{\theta}, x^{(j)}\big) - E^{\mathrm{qm}}\big(x^{(j)}\big) \Big)^2 + C_c \sum_{i=1}^{m} \boldsymbol{\theta}_i^2, \tag{19}$$

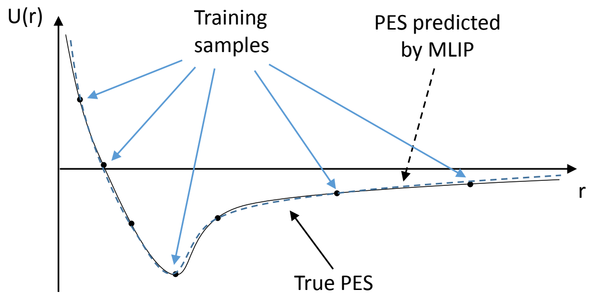
where N is the size of the training set, $x^{(j)}$ are the configurations in the training set, $E^{\mathrm{qm}}\big(x^{(j)}\big)$ are their reference energies, and $C_c \sum_{i=1}^{m} \boldsymbol{\theta}_i^2$ is the so-



(a)



(b)



(c)

Figure 5: Fitting of Lennard-Jones potential with MLIP. In cases (a) and (b) the training samples do not cover some regions of the configurational space, which leads to incorrect form of the reproduced PES in this regions. When the interior of the configurational space is covered evenly (case (c)) the form of the PES is reproduced correctly.

called *soft constraint* term, introduced to distinguish different but equivalent solutions based on their norm (see Section 3.5 for details). In a case of fitting not only to the energies, but also to the forces and stresses, the loss function is extended with additional summands:

$$\begin{aligned} L(\boldsymbol{\theta}) =& C_e \sum_{j=1}^{N} \Big(E\big(\boldsymbol{\theta}, x^{(j)}\big) - E^{\text{qm}}\big(x^{(j)}\big)\Big)^2 + C_f \sum_{j=1}^{N} \sum_{k=1}^{n_j} \Big(\boldsymbol{f}_k\big(\boldsymbol{\theta}, x^{(j)}\big) - \boldsymbol{f}_k^{\text{qm}}\big(x^{(j)}\big)\Big)^2 \\ & + C_s \sum_{j=1}^{N} \sum_{a=1}^{3} \sum_{b=1}^{3} \Big(\sigma_{ab}\big(\boldsymbol{\theta}, x^{(j)}\big) - \sigma_{ab}^{\text{qm}}\big(x^{(j)}\big)\Big)^2 + C_c \sum_{i=1}^{m} \boldsymbol{\theta}_i^2, \end{aligned} \tag{20}$$

where index k enumerates the atoms in the configuration j of size n_j; $\boldsymbol{f}_k\big(\boldsymbol{\theta}, x^{(j)}\big)$ and $\boldsymbol{f}_k^{\text{qm}}\big(x^{(j)}\big)$ are the predicted and the reference forces acting on atom k, $\sigma_a\big(\boldsymbol{\theta}, x^{(j)}\big)$ and $\sigma_{ab}^{\text{qm}}\big(x^{(j)}\big)$ are predicted and reference virial stress components. Multipliers C_e, C_f, and C_s determine the relative weights of energies, forces, and stresses in the loss function and, consequently, in the fitting procedure. Their typical values for the case of MTP fitting (see Section 3.1) values are:

$$\begin{aligned} C_e &= 1, \\ C_f &= 10^{-2} \mathring{A}^2, \\ C_s &= 10^{-3} \mathring{A}^6, \end{aligned} \tag{21}$$

while typical values for C_c are in the $10^{-8} - 10^{-6} eV^2$ range. The numbers from (21) come from practice and reflect the condition of optimal relative importance of the energy, forces, and stresses factors in the fitting procedure. I.e., too small or too large value of any coefficient from (21) will result in neglecting/overprevailing of a certain factor among the others.

The expression for the energy $E^{(j)}$ of configuration $x^{(j)}$ is composed from the contributions of individual atomic neighborhoods:

$$E^{(j)} = E\big(\boldsymbol{\theta}, x^{(j)}\big) = \sum_{i}^{n_j} V(\mathfrak{n}_i^j), \tag{22}$$

where index i enumerates the atoms of the configuration $x^{(j)}$ and $\mathfrak{n}_i^j$ denote their neighborhoods. Using the partitioning scheme for energy (22) forces acting on atoms are calculated as:

$$\boldsymbol{f}_k\big(\boldsymbol{\theta}, x^{(j)}\big) = -\frac{\partial E\big(\boldsymbol{\theta}, x^{(j)}\big)}{\partial \boldsymbol{r}_{kj}} = -\sum_{i=1}^{n_j} \frac{\partial V(\boldsymbol{\theta}, \mathfrak{n}_i^j)}{\partial \boldsymbol{r}_{kj}}, \tag{23}$$

where $\boldsymbol{r}_{kj}$ is the position of atom k in configuration j. Virial stress in an atomistic configuration with the lattice volume Ω is given by:

$$\sigma_{ab} = \frac{1}{2\Omega} \sum_{k,l=1}^{n_j} (x_a^l - x_a^k) f_b^{kl},$$

where f_b^{kl} is the b component of the force applied on atom k by atom l. As is shown further, the function V is differentiable with respect to atomic coordinates, thus forces and stresses can be calculated for any set of parameters $\boldsymbol{\theta}$ and for any configuration.

3.4 Moment Tensor Potentials

Mathematically, each atom in the neighborhood introduces four degrees of freedom, on which $\mathfrak{n}_i$ depends: three continuous coordinates in Euclidean space, and a discrete variable representing the type of the atom. Typically, neighborhoods include few dozen atoms, which means that the function $V(\mathfrak{n}_i)$ depends on the order of hundred scalar variables. To somewhat reduce the dimensionality, all physical symmetries are embedded into $V(\mathfrak{n})$ so they will not have to be learned by the model. These symmetries arise from isotropy and translational symmetry of the physical space, and from the fact that the interaction between atoms does not depend on their ordering.

As in the work [76] devoted to the single-component moment tensor potentials, $V(\mathfrak{n})$ is linearly expanded through a set of *basis functions* B_α:

$$V(\mathfrak{n}) = \sum_\alpha \xi_\alpha B_\alpha(\mathfrak{n}). \tag{24}$$

The basis functions, in turn, depend on the set of *moment tensor descriptors*

$$M_{\mu,\nu}(\mathfrak{n}_i) = \sum_j f_\mu(|\boldsymbol{r}_{ij}|, z_i, z_j)\, \underbrace{\boldsymbol{r}_{ij} \otimes ... \otimes \boldsymbol{r}_{ij}}_{\nu \text{ times}}, \tag{25}$$

where the index j enumerates all the atoms in the neighborhood $\mathfrak{n}_i$.

To define the basis functions, we choose a cut-off radius R_{cut} and introduce, as described below, a representation of the neighborhoods which is invariant with respect to rotations and permutations of chemically equivalent atoms. Note that the translation invariance is already built into (2). The symbol "$\otimes$" stands for the outer product of vectors, thus in (25) $\boldsymbol{r}_{ij} \otimes ... \otimes \boldsymbol{r}_{ij}$ is the tensor of rank ν. This way, each descriptor in (25) is composed of the radial part $f_\mu(|\boldsymbol{r}_{ij}|, z_i, z_j)$ which depends only on the relative distances between atoms and on their types and on the angular part $\boldsymbol{r}_{ij} \otimes ... \otimes \boldsymbol{r}_{ij}$ resembling the moments of inertia. It should be emphasized that $M_{\mu,0}$ are the standard two-body descriptors of atomic environments that do not contain information about angles between bonds. The general moment tensor descriptors $M_{\mu,\nu}$ remain two-body and thus offer an alternative way of including the angular information—the traditional way is to include at least three-body descriptors. The functions $f_\mu(|\boldsymbol{r}_{ij}|, z_i, z_j)$ depend only on the interatomic distances and atomic types, therefore we call them *radial functions.* The terms $\boldsymbol{r}_{ij} \otimes ... \otimes \boldsymbol{r}_{ij}$ contain the angular information about the neighborhood $\mathfrak{n}_i$.

We next explain how to construct the basis functions from the moment tensor descriptors, following which we present a simple illustration of the structure of the descriptors and basis functions. The functions $B_\alpha(\mathfrak{n}_i)$ enumerate all possible contractions of any number of $M_{\mu,\nu}(\mathfrak{n}_i)$ yielding a scalar. Note that $M_{\mu,\nu}(\mathfrak{n}_i)$ are invariant, by construction, with respect to translations

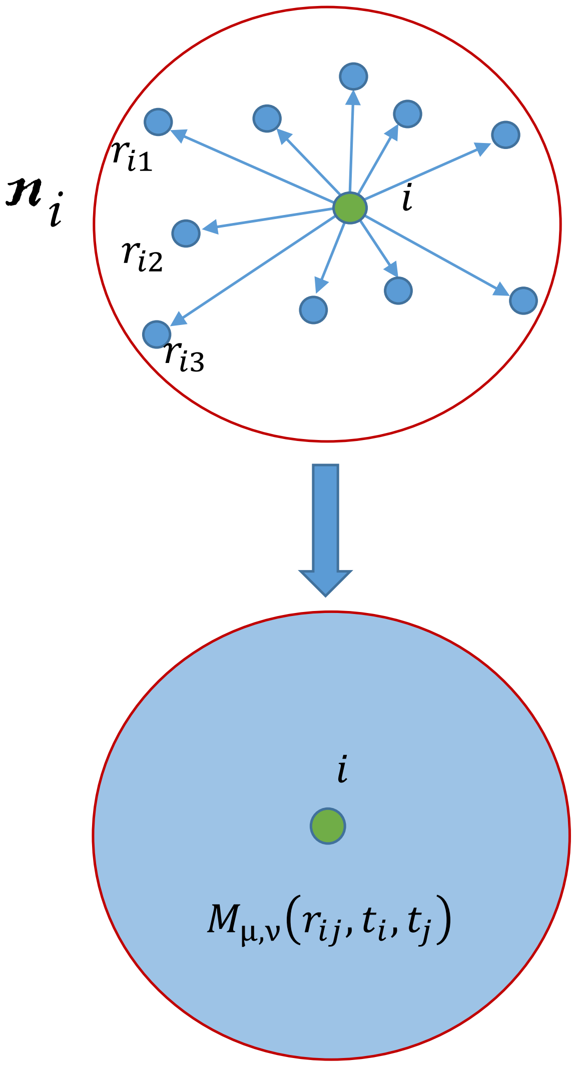


Figure 6: For the purpose of fitting the interatomic interaction energy E, the neighborhood $\mathfrak{n}_i$ is described by the moment tensors $M_{\mu,\nu}$ exhibiting all the physical symmetries that E has. The descriptors $M_{\mu,\nu}$ depend on distances $\boldsymbol{r}_{ij}$ and chemical types t_i, t_j in all pairs of atoms in the neighborhood, including the central i and the peripheral ones j.

of the system and permutations of equivalent atoms. Their scalar contractions are invariant with respect to rotations of the neighborhood. Thus the resulting function $V(\mathfrak{n})$ also has these symmetries. Although all the descriptors $M_{\mu,\nu}(\mathfrak{n}_i)$ are composed of two-body terms depending only on $\boldsymbol{r}_{ij}$, their contractions $B(\mathfrak{n}_i)$ can depend on many-body terms of higher order.

For the purpose of illustration assume, for the moment, that the vectors $\boldsymbol{r}_{ij}$ are two-dimensional and that they are expressed in polar coordinates (ρ, θ) centered at the ith atom. Let us look closer at the term $\boldsymbol{r}_{ij} \otimes \ldots \otimes \boldsymbol{r}_{ij} \equiv r^{\otimes\nu}$, which is a tensor of rank ν. E.g., $\boldsymbol{r}_{ij}^{\otimes 0}$ is a scalar with no angular information, while $\boldsymbol{r}_{ij}^{\otimes 1} = \boldsymbol{r}_{ij} = |\boldsymbol{r}_{ij}|(\cos\theta_{ij}, \sin\theta_{ij})$ does contain angular information. A vectorial contraction is simply a dot product: $\boldsymbol{r}_{ij} \cdot \boldsymbol{r}_{ik} = |\boldsymbol{r}_{ij}|\,|\boldsymbol{r}_{ik}|\cos(\theta_{ij} - \theta_{ik})$—in this way angular terms are introduced into the potential. An arbitrary function of angle can be expanded into a sum of powers of cosine. Such higher-order terms are contributed to the potential by higher-rank tensors, e.g.,

$$\boldsymbol{r}_{ij}^{\otimes 2} = \boldsymbol{r}_{ij}\boldsymbol{r}_{ij}^{\top} = |\boldsymbol{r}_{ij}|^2 \begin{pmatrix} \cos^2\theta_{ij} & \sin\theta_{ij}\cos\theta_{ij} \\ \sin\theta_{ij}\cos\theta_{ij} & \sin^2\theta_{ij} \end{pmatrix}.$$

The contractions of two matrices are given by the Frobenius product

$$\boldsymbol{r}_{ij}^{\otimes 2} : \boldsymbol{r}_{ik}^{\otimes 2} = |\boldsymbol{r}_{ij}|^2 |\boldsymbol{r}_{ik}|^2 \cos^2(\theta_{ij} - \theta_{ik}).$$

A more complicated expression can be constructed with a matrix and two vectors:

$$(\boldsymbol{r}_{ij}^{\otimes 2}\boldsymbol{r}_{ik}) \cdot \boldsymbol{r}_{i\ell} = |\boldsymbol{r}_{ij}|^2 |\boldsymbol{r}_{ik}||\boldsymbol{r}_{i\ell}| \cos(\theta_{ij} - \theta_{ik})\cos(\theta_{ij} - \theta_{i\ell}).$$

Terms of this form are rotationally invariant. Permutation invariance is achieved by summing those terms over all atoms in the neighborhood weighted by the radial functions.

As an illustration, assume that there are two radial functions,

$$f_\mu(\rho, z_i, z_j) = \exp\Big(-\frac{|\rho - R_\mu|^2}{2\sigma^2}\Big),$$

$\mu = 1, 2$, where ρ has the meaning of distance to the central, ith atom. In the sum (25) they "extract" two shells of atoms, around the distances R_1 and R_2 from the ith atom, smeared over the width of σ. We did not, but could assume the dependence of these functions on the types of atoms z_i and z_j—this would discriminate the importance of these atom types to these two shells. Thus, $M_{1,0}$ and $M_{2,0}$ are the atom count in these two shells and both can serve as basis functions. $M_{i,1}$ are vectorial quantities indicating eccentricity of these shells: if $M_{i,1} = 0$ then the ith shell is symmetric (to the first order) while $M_{i,1} \neq 0$ indicate that there are "more atoms" in the direction $M_{i,1}$ than in the opposite direction.

As vectorial quantities, $M_{i,1}$ are not the valid basis functions, however, the valid ones are $M_{i,1} \cdot M_{i,1}$ indicating the magnitude of eccentricity and $M_{i,1} \cdot M_{i,2}$ indicating how these two eccentricities are aligned with respect to each other. One can make many more basis functions from these quantities, e.g., $M_{i,0}(M_{i,1} \cdot M_{i,1})$, $(M_{i,1} \cdot M_{i,1})(M_{i,1} \cdot M_{i,2})$, etc. One can then continue by analogy: $M_{i,2}$ are the second moments of inertia of these shells indicating the degree to which these shells are "squeezed" in the respective directions, forming the basis functions $M_{i,2} : M_{j,2}$, $(M_{i,2} M_{j,1}) \cdot M_{k,1}$, $(M_{i,2} M_{j,2} M_{k,1}) \cdot M_{\ell,1}$, etc. We remark that this way of enforcing symmetries in the potential is related to the ideas from Refs.[46, 38].

For the purpose of choosing which (out of the infinite number of) basis functions to include in the interatomic potential, we define the degree-like measure, *level*, of $M_{\mu,\nu}$ by $\mathrm{lev} M_{\mu,\nu} = 2\mu + \nu$ and the level of B_α obtained by contracting $M_{\mu_1,\nu_1}, M_{\mu_2,\nu_2}, \ldots$, as $\mathrm{lev} B_\alpha = (2\mu_1 + \nu_1) + (2\mu_2 + \nu_2) + \ldots$. Thus, to define an MTP we choose some $\mathrm{lev}_{\max}$ and include in (24) each B_α with $\mathrm{lev} B_\alpha \leq \mathrm{lev}_{\max}$. Thus, by increasing $\mathrm{lev}_{\max}$ one can increase the number of parameters in the potential, including the contributions of three-body, four-body, etc., terms. In this sense, $V(\mathfrak{n})$ has a systematically improvable functional form. By increasing $\mathrm{lev}_{\max}$ models with more parameters are emerged, capable of more accurate fitting but requiring more computational time and more data to train. Also the calculations of energy, forces, and stresses are performed slowly for MTPs with higher $\mathrm{lev}_{\max}$.

The radial functions $f_\mu(|\boldsymbol{r}_{ij}|, z_i, z_j)$ from (25) have the form:

$$\begin{aligned} f_\mu(\rho, z_i, z_j) &= \sum_k c^{(k)}_{\mu,z_i,z_j} Q^{(k)}(\rho), \qquad \text{where} \\ Q^{(k)}(\rho) &:= T_k(\rho)(R_{\mathrm{cut}} - \rho)^2. \end{aligned} \tag{26}$$

Here $T_k(\rho)$ are the Chebyshev polynomials on the interval $[R_{\min}, R_{\mathrm{cut}}]$. The term $(R_{\mathrm{cut}} - \rho)^2$ was introduced to ensure smoothness with respect to the atoms leaving and entering the cut-off sphere. Taking into account that in real systems atoms never stay too close to each other, it is always possible to choose some reasonable value for $R_{\min}$.

The difference from the single-component MTPs [76] is that now the functions $f_\mu(\rho, z_i, z_j)$ depend on the types of the central and the neighboring atoms. The graphical scheme of the $V(\mathfrak{n})$ function is illustrated in the Figure 7. Note, that $V(\mathfrak{n})$ is linear with respect to parameters $\{\xi_\alpha\}$, while the parameters $\{c^{(k)}_{\mu,z_i,z_j}\}$ propagate through several layers of calculations: $f_\mu \to M_{\mu,\nu} \to B_\alpha \to V$ and resemble the deep layers weights in neural networks.

As follows from (26) a number of parameters $c^{(k)}_{\mu,z_i,z_j}$ exist for each pair of species and each μ. Note that the number of these parameters is proportional to n^2, where n is the number of species, while number of parameters ξ_α from (24) does not depend on the number of species. Thus, the total number of model parameters $\boldsymbol{\theta} = (\{\xi_\alpha\}, \{c^{(k)}_{\mu,z_i,z_j}\})$ to be found in the minimization procedure (19) grows less than quadratically with the number of species, despite accounting for many-body interactions in $V(\mathfrak{n})$. It was proven [76] that the descriptors of the form (25) provide a complete description of an atomic neighborhood, in a sense that any function of atomic neighborhood with the same symmetries as $V(\mathfrak{n})$ can be approximated as a polynomial of these descriptors with an arbitrary accuracy. The proof [76] holds only for a single-component case. While for a multicomponent case this may not necessarily be true, the introduced nonlinear parameterization (26) of the new degrees of freedom (namely, the atomic types) with polynomial approximators is still an approach, which further (Section 4) is shown to be worthwhile.

Despite the magnetic moments of the atoms are neglected in the current implementation, they can be included in the current implementation by adding a dependence of the radial functions $f_\mu(\rho, z_i, z_j)$ from (25) on the magnetic moments of atoms in each pair participating in the tensor moment construction. The below formula provides generalized radial functions expression for a case of collinear magnetic moments:

$$f_\mu(\rho, z_i, z_j, m_i, m_j) = \sum_k c^{(k)}_{\mu,z_i,z_j} Q^{(k)}(\rho) \sum_{k_1,k_2} c^{(k,k_1,k_2)}_{\mu,z_i,z_j} Q^{(k_1)}(m_i) Q^{(k_2)}(m_j), \tag{27}$$

$$\text{where} \tag{28}$$

$$Q^{(k)}(\rho) := T_k(\rho)(R_{\text{cut}} - \rho)^2.$$

In this implementation the magnetic moments of atoms are treated as independent variables on which the potential energy depends. This adds two dimensions (as radial functions are constructed for pairs of atoms) to a space of independent variables on which $f_\mu(\rho, z_i, z_j, m_i, m_j)$ depends, and two additional sums in the expansion of this function through polynomials $Q^{(k)}(\rho)$. In a general case of 3-d magnetic moments, it will be 6 additional dimensions and 6 sums over corresponding indexes.

The only means by which the MTPs can describe charged systems are screening of electrostatic interaction, which makes the locality assumption (2) applicable. In the case of essentially non-local electrostatic interaction the charges should be included explicitly within hypothetical non-local implementation.

3.5 Training the Moment Tensor Potentials

While training of a single-component MTP means solving the overdetermined system of linear algebraic equations (a single-component MTP is linear w.r.t its parameters [76]), which is purely procedural and straightforward task, training a multicomponent MTP requires solving the non-linear optimization

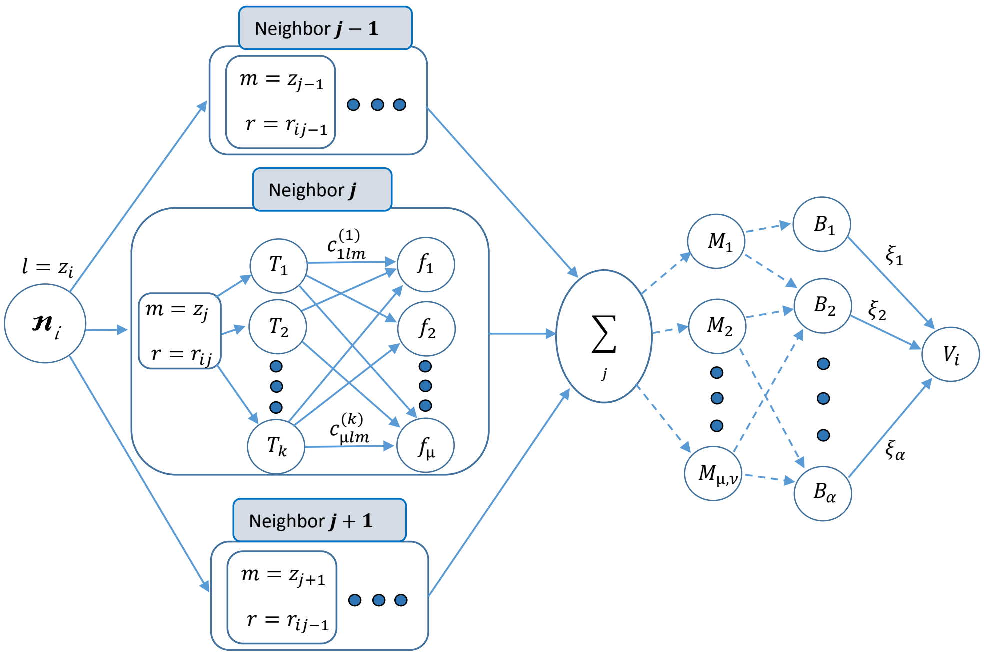


Figure 7: Computational scheme of the moment tensor potentials. The number of basis functions B_α (and respective coefficients ξ_α) is a tunable hyperparameter. The dependence of V_i on the types of atoms in $\mathfrak{n}_i$ is incorporated in radial functions $f_\mu(\rho, z_i, z_j)$.

problem in high-dimensional space. For being able to use the gradient-based optimization methods like gradient descent one needs to compute the first (at least) derivatives of the loss function (19) w.r. its parameters, which takes much more computational effort then calculating just a loss function value. Therefore I have implemented a back-propagation algorithm for derivatives calculation which calculates all the loss function derivatives w.r.t MTP coefficients (which are hundreds) only 4.8 times slower than the actual loss function calculation.

Another important question arising during non-linear optimization is choosing the optimization method. I have tried both simulated annealing and basin-hopping techniques of global optimization (see Section 3.1) together with BFGS to find several solutions of the problem (15). It turned out that while training of MTPs, different solutions of the problem resulted in the same accuracy, and the difference in the parameters $\boldsymbol{\theta}$ was caused only by inner symmetries of the model, i.e., existence of different parameter sets $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_1$, such as $F(\boldsymbol{\theta}_1) = F(\boldsymbol{\theta}_2)$ and, consequently $L(\boldsymbol{\theta}_1) = L(\boldsymbol{\theta}_2)$. It is worth to note, that finding a local minimum consumes some amount of computational time, and finding several local minima requires several times more, as the local optimization algorithm is launched several times. In such circumstances finding several local minima providing the same accuracy is a waste of computational time.

Therefore, I ended up with just one local optimization as the training procedure. However, symmetries present in the model can (and actually do) hamper the convergence of local optimization algorithms like BFGS, as there can be back and forth stepping along the direction between two identical solutions. To get rid of identical solutions we introduced an artificial penalty which results in preferring some solutions against others, thus allowing the optimization algorithm to converge more unequivocally:

$$L(\boldsymbol{\theta}) = \sum_{j=1}^{n} \big(y_j - F\big(\boldsymbol{\theta}, x^{(j)}\big)\big)^2 + C \sum_{i=1}^{m} \boldsymbol{\theta}_i^2, \tag{29}$$

where C is some small positive number.

Since m is the only tunable hyperparameter of our model (more details in Section 3.4), our testing has found that a good rule of thumb for choosing it is $n \geqslant 2m$.

3.6 Active Learning

While constructing a machine-learning potential the problem of sampling arises (see Section 3.2): to run an MD simulation of some process of interest (e.g., a molecular reaction or a phase transition) one should have training set of representative configurations relevant to a certain simulation. In order to sample such configurations, one needs a potential visiting proper parts of the configurational space. This, in turn, can be done with the potential already trained on representative configurations. While traditional iterative resampling-retraining approach does not offer systematic success criterion and is not always efficient, we propose the *active learning* approach, which makes possible to sample the configurational space and run a simulation at the same time. The approach we propose is based on the so-called *D-optimality* criterion, which provides a numerical criterion for "novelty" of a certain configuration for interatomic potential (namely, MTP). This essentially means proximity of the configuration to the training set of MTP. MTP "extrapolates" when configuration does not belong to the interior of the configurational space, covered by the training set. Definition and detailed description of the extrapolation grade concept is given in section 3.6.1.

Depending on the problem to be solved, active learning can be implemented in a different ways. In this thesis two scenarios of active learning usage are provided: active learning with validation set (Section 3.6.2)and active learning in crystal structure prediction (Section 3.6.3).

3.6.1 Generalized D-optimality Criterion

Recently the active learning approach has been proposed, designed for interatomic potentials with a linear dependence on the model parameters [60]. This approach is based on a D-optimality criterion for selecting the training dataset, which is equivalent to choosing atomistic configurations maximizing the determinant of the matrix of the linear equations on the model parameters. The model proposed in this paper has a nonlinear dependence on its parameters $\boldsymbol{\theta}$, therefore we propose a generalization of the D-optimality criterion to the nonlinear case. To that end, we assume that the values of the parameters $\bar{\boldsymbol{\theta}}$, which are found from the training procedure (Section 3.1) are already near the optimal ones and we hence linearize each term in the function (15) with respect to the parameters:

$$y^{(i)} - F\big(\boldsymbol{\theta}, x^{(i)}\big) \approx y^{(i)} - \sum_j (\theta_j - \bar{\theta}_j)\,\frac{\partial F}{\partial \theta_j}\big(\bar{\boldsymbol{\theta}}, x^{(i)}\big).$$

One can then interpret the fitting as the solution of the following overdetermined system of equations with respect to θ_j:

$$\sum_{j=1}^{m} \theta_j \frac{\partial F}{\partial \theta_j}\big(\bar{\boldsymbol{\theta}}, x^{(i)}\big) = y^{(i)} + \bar{\theta}_j \frac{\partial F}{\partial \theta_j}\big(\bar{\boldsymbol{\theta}}, x^{(i)}\big),$$

$$i = 1, \ldots, n,$$

where n is the size of the training set $\{x^{(1)}, \ldots, x^{(n)}\}$. The matrix of this system is a tall Jacobi matrix

$$\mathsf{B} = \begin{pmatrix} \frac{\partial F}{\partial \theta_1}\big(\bar{\boldsymbol{\theta}}, x^{(1)}\big) & \ldots & \frac{\partial F}{\partial \theta_m}\big(\bar{\boldsymbol{\theta}}, x^{(1)}\big) \\ \vdots & \ddots & \vdots \\ \frac{\partial F}{\partial \theta_1}\big(\bar{\boldsymbol{\theta}}, x^{(n)}\big) & \ldots & \frac{\partial F}{\partial \theta_m}\big(\bar{\boldsymbol{\theta}}, x^{(n)}\big) \end{pmatrix},$$

where each row corresponds to a particular configuration from the training set.

Next, a subset of configurations yielding the most linearly independent rows in B is selected for training. This is equivalent to finding a square $m \times m$ submatrix A of maximal volume (i.e., with maximal value of $|\det(\mathsf{A})|$). We have implemented it by using the so-called MaxVol algorithm [30].

We define the "novelty" grade (extrapolation grade) $\gamma(x^*)$ as the maximal factor by which $|\det(\mathsf{A})|$ can grow if x^* is added to the training set. According to Ref.[30] it can be calculated as

$$\gamma(x^*) = \max_{1 \le j \le n} (|c_j|),$$

where

$$c = \left(\frac{\partial F}{\partial \theta_1}(\bar{\boldsymbol{\theta}}, x^*) \dots \frac{\partial F}{\partial \theta_n}(\bar{\boldsymbol{\theta}}, x^*) \right) \mathsf{A}^{-1} =: b^* \mathsf{A}^{-1}.$$

Thus, if $\gamma(x^*) \geq 1$ the configuration contains new information and can be included into the training set. In practice, some threshold $\gamma_{trsh} > 1$ is used to prevent configuration with not sufficiently high extrapolation grade $\gamma(x^*) < \gamma_{trsh}$ from entering the training set. The principle of sampling the configurations based on their extrapolation grade γ is further embedded into the active learning approaches, used in different simulation scenarios.

Obviously, to some extent the extrapolation grade correlates with the error which MTP is expected to have for a certain configuration. Though, the authors emphasize that the essence of this criterion is measure of proximity of configuration to the training domain of certain MTP. Nevertheless, it would be illustrative to provide such dependence for a case of MTP designed for a TiN binary system 8. The MTP used for this test has 50 parameters and configurations are taken from NPT simulations of B1 TiN at 300K.

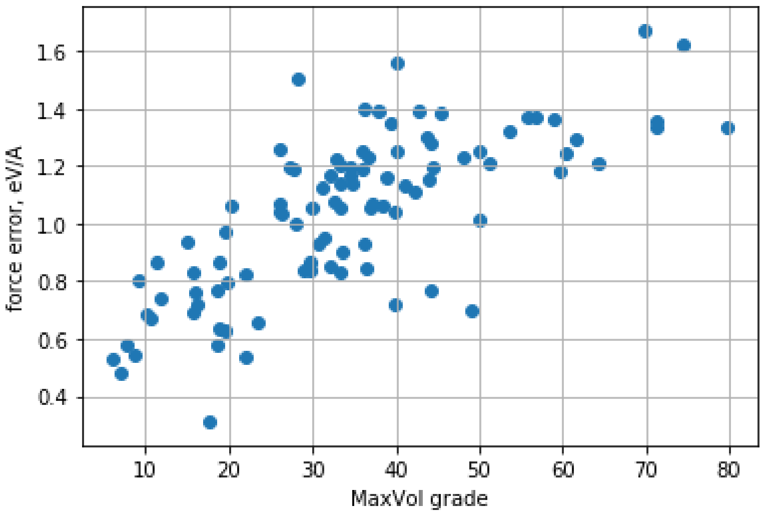


Figure 8: The illustration of a connection between extrapolation grade and the absolute force error of MTP.

3.6.2 Active Learning with Validation Set

The problem solved in this case can be formulated in a following way: given a set of configurations (e.g., molecules or periodic structures) and a MLIP with fixed functional form (but non-fixed coefficients), it is needed to select a subset of configurations by training on which the MLIP will exhibit the best accuracy of predicting the properties (e.g., formation energy) of the remaining configurations (called validation set in this case).

On the one hand, the training set should represent the full variety of configurations to prevent extrapolation while evaluating properties of configurations from the validation set. On the other hand, the size of the training set is typically limited by the amount of experiments or *ab initio* calculations can be conducted in a reasonable time. It has been shown [40] that the optimal choice of the training set of a fixed size can, in principle, significantly reduce the validation errors, if one was allowed to use the labels $y^{(i)}$ (see Section 3.1) of all the available data. However, a practical selection algorithm needs to choose configurations for the training set based only on the unlabeled data $x^{(i)}$ (as our proposed algorithm does), since one wants to compute the labels only after selection.

The active learning algorithm we propose effectively detects configurations on which the MLIP extrapolates. Hence, training on such configurations prevents extrapolation and thus ensures reliable treatment of the remaining configurations at the evaluation stage. We next describe our active learning procedure applied to a problem of selecting the best subset for training from some pre-defined set. This procedure forms the training set iteratively, each time increasing its size by not more than 10%. The steps are the following:

0. Start with a random initial training set of a small size.
1. Train the model on the current training set.
2. Using the active learning algorithm [60] select configurations with $\gamma \geq \gamma_{\rm trsh}$, and add either all such configurations or the configurations with

the highest γs, such that the size of the training set increases by 10% or less.

3. Unless satisfied with the current model (see the discussion below), go to the step 1.

The loop stops (as mentioned in the step 3) when the difference in accuracy of the models on several consecutive iterations is too small. We have observed that sometimes the accuracy improvement on one particular iteration may be small, while on the next one it can increase again. Therefore, tracking improvement on several iterations of the cycle is more reliable criterion.

3.6.3 Active Learning in Crystal Structure Prediction

Crystal structure prediction (CSP) is aimed at searching the most stable structures for a given atomic composition or a range of compositions. Methods for crystal structure prediction involve evaluation of energies for large amounts of different structures and selecting the most stable ones (with the lowest formation energies). The structures to be checked are typically provided by some generative algorithm [35, 52] or are selected from some pool of structures [54] containing a diverse set of geometries and compositions, the more the better. Applying MLIP to CSP yields similar problems to using MLIP in MD: MLIP should predict properties of configurations not present in a training set, with a chance of extrapolation resulting in an inaccurate prediction for a given structure. Using the active learning approach it is possible to perform a simulation in which evaluation of the occurring structures and their sampling for the training set is done simultaneously.

In a crystal structure prediction one needs to perform relaxations (see Figure 11) of different crystal structures with MLIP. Therefore, the training set should include all the representative structures, so that the potential does not have to extrapolate while searching for the stable structures. In cluster expansion-like approaches in which the energies of the relaxed structures are predicted based on representation that uses unrelaxed structure geometries, extrapolation results in higher prediction errors [58]. However, in our approach structural relaxation is performed explicitly, and we accelerate the relaxation process by using a machine-learning potential instead of DFT, see the scheme Figure 9. Because of the added flexibility of the MTP (as compared to cluster expansion), avoiding extrapolation is even more important—it is crucial to the reliability of the algorithm—as highly unphysical structures can arise during the relaxation if the extrapolation is severe.

Here we describe a special case of crystal structure prediction—construction of the convex hull for metallic alloys. Convex hull is a geometrical surface in a compositional space (see Section 4.1 for details) containing the stable structures for all possible concentrations of constituting elements. Next we describe the algorithm for constructing a convex hull using the active learning approach.

It is assumed that the starting training set is empty and is only composed algorithmically as is described below. This allows for excluding the human factor on this stage and therefore making the procedure more generic and more automatized. One can, however, still start from some initial set of configurations to end the procedure faster with some final training set as the result (which is still more or less the same regardless of the initial training set). To benefit from the active learning approach one should not provide too much configurations from the beginning, to leave the space for new incoming configurations. While the size of the final training set would be about $2m$, where m is the amount of parameters in the MTP, we recommend to take not more than roughly $m/2$ configurations for initialization.

Input The input to the algorithm is:

1. A set of candidate structures among which one expects to find the ground state structures. It is possible to select much broader and more diverse set of structures as compared to the approaches based solely on DFT. Note, that they will change their geometries during relaxation, but not compositions.

2. A functional form of MTP, $E = E(\boldsymbol{\theta}, x)$.

   Initialize $\boldsymbol{\theta}$ randomly and let the training set be empty.

3. A quantum-mechanical model $E^{\mathrm{qm}}(x)$.

   In this thesis it means DFT as implemented in VASP 5.4.1.

4. Two thresholds γ_{tsh} and Γ_{tsh}, such that $\Gamma_{\mathrm{tsh}} > \gamma_{\mathrm{tsh}} > 1$.

   If the extrapolation grade $\gamma(x^*)$ is greater than 1, the algorithm makes two decisions: to add x^* to the training set if $\gamma(x^*) > \gamma_{\mathrm{tsh}}$ and to terminate the relaxation if $\gamma(x^*) > \Gamma_{\mathrm{tsh}}$ (assuming in the latter case that MTP cannot make reliable predictions of energy, forces, and stresses for x^*), as explained below.

Step 1 For each candidate structure perform structure relaxation with the current MTP (defined by the current values of $\boldsymbol{\theta}$). There can be two outcomes of the relaxation: (1) the relaxation is completed successfully and equilibrium structure appears as a result, (2) the relaxation was not successful because a structure emerged on which the MTP attempted to extrapolate. More precisely, the following scenarios can emerge:

a. The relaxation successfully converges to an equilibrium configuration and on each configuration from the relaxation trajectory the MTP does not significantly extrapolate, i.e., the extrapolation grade of each intermediate configuration is less than Γ_{tsh}. During the relaxation there could be, however, configuration with extrapolation grade exceeding γ_{tsh}—in this case such a configuration is added to the *preselected set* (see Figure 9 and Section 3).

b. At some step of the relaxation a configuration with extrapolation grade exceeding Γ_{tsh} is emerged. This means that MTP cannot provide a reasonable prediction as it extrapolates significantly on this configuration and needs to be retrained with more *ab initio* data. The relaxation is then terminated. The last and all the previous configurations with the grade exceeding γ_{tsh} are added to the *preselected set*.

Step 2 Out of the preselected set from the step 1b, select a smaller number of configurations that will be added to the training set. The preselected set can be very large and contain hundreds of thousands configurations (note that during the first iteration of the algorithm all the relaxations will be terminated according to the scenario (b), as the training set is empty and the MTP extrapolates on every configuration). Therefore the active learning algorithm is used to select up to few hundred most representative configurations, according to the D-optimality criterion from Section 3.6.1. It extends the training domain of the MTP as much as possible while keeping the amount of *ab initio* calculations relatively small. After the calculation, *ab initio* energies, forces, and stresses of the selected configurations are added to the training set.

Step 3 Fit the MTP to the updated training set. As the size of the training set grows with each iteration of the algorithm, this step will take more and more time on each subsequent iteration, but still this time is a small fraction of the time spent on *ab initio* calculations.

Step 4 Repeat the steps 1–3 unless all the relaxations have successfully converged to the respective equilibrium configurations.

As the MTP is repeatedly refitted during the relaxation on a dynamically updated training set, we call this algorithm as "relaxation while learning on-the-fly".

3.7 Discussion

In this section the detailed description of the research methodology of the present thesis was provided. There were introduced: MTPs (a special type of MLIPs), the active learning approach (the way of automatic composition of the training set which prevents MLIP extrapolation during simulation), and the nuances of MLIPs training.

MTPs cannot predict the long-range ordering in materials as they are essentially local (see Section 2.1). Due to the same reason MTPs by their

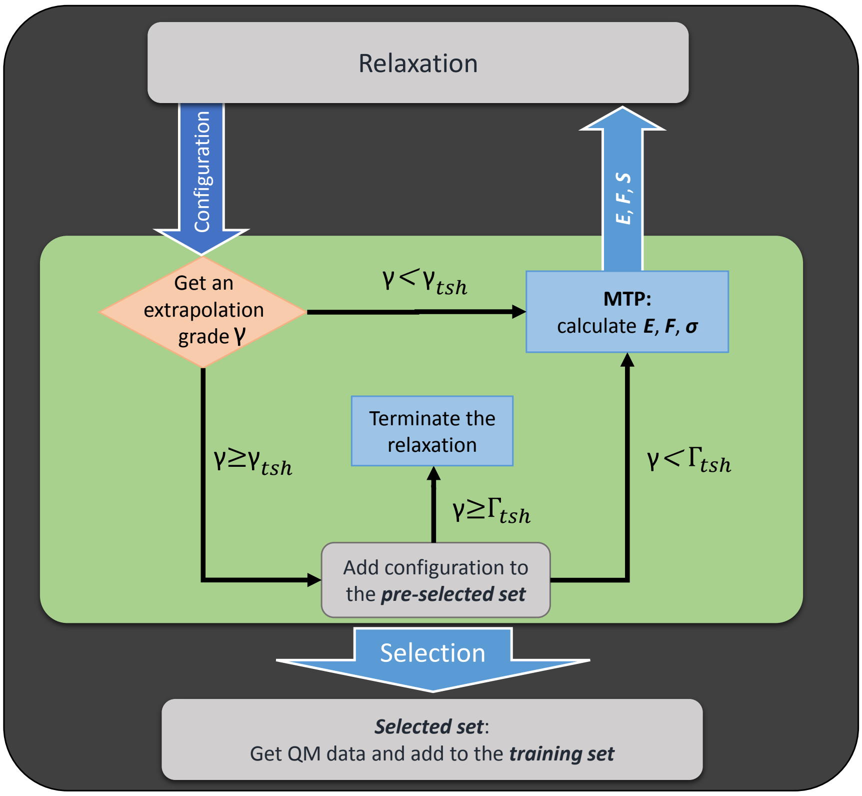
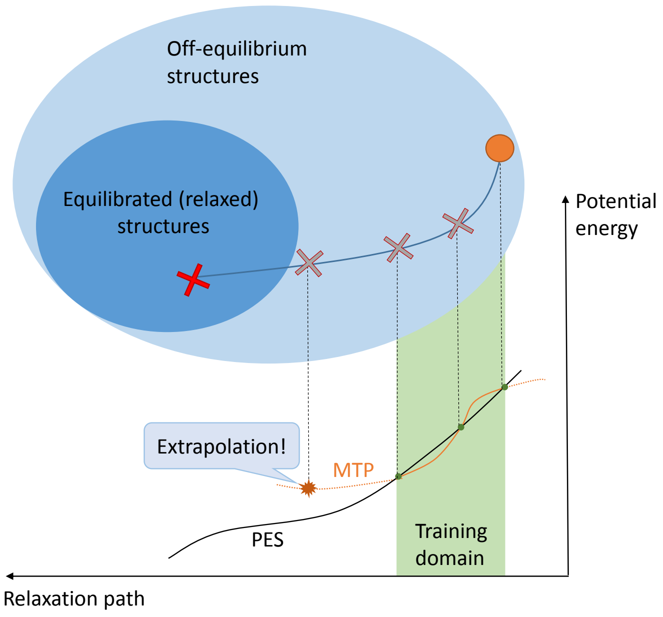
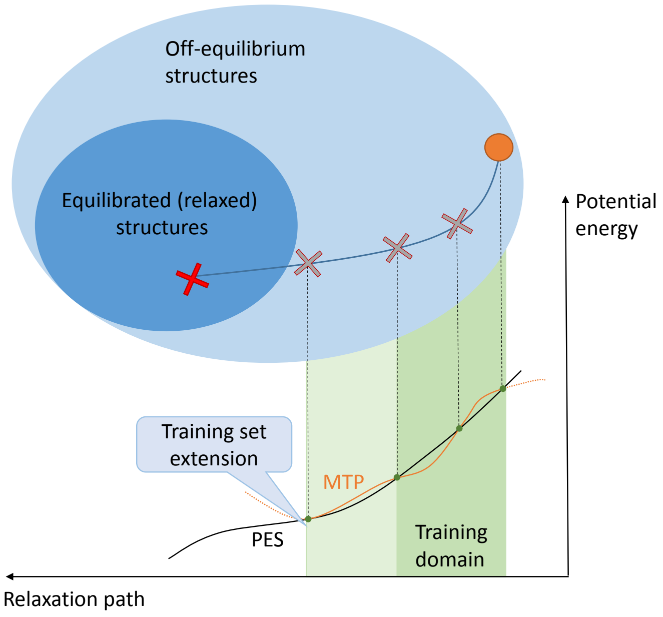

Relaxation
Configuration
E, F, S
Get an extrapolation grade γ
γ<γ_tsh
MTP: calculate E, F, σ
Terminate the relaxation
γ≥γ_tsh
γ<Γ_tsh
γ≥Γ_tsh
Add configuration to the pre-selected set
Selection
Selected set: Get QM data and add to the training set

Figure 9: Relaxation with active learning. If MTP encounters an extrapolative configuration ($\gamma \geq \gamma_{\mathrm{tsh}}$), this configuration is added to the preselected set for further selection. In the case of significant extrapolation ($\gamma \geq \Gamma_{\mathrm{tsh}}$) the relaxation is terminated. For configurations with $\gamma < \Gamma_{\mathrm{tsh}}$, the MTP provides energies, forces and stresses. If no configuration with $\gamma \geq \Gamma_{\mathrm{tsh}}$ is encountered, the relaxation stops at some equilibrium configuration.



(a)



(b)

Figure 10: If an MTP encounters some extrapolative configuration during relaxation, as shown in (a), the relaxation is terminated and restarted after retraining the MTP, as shown in (b).

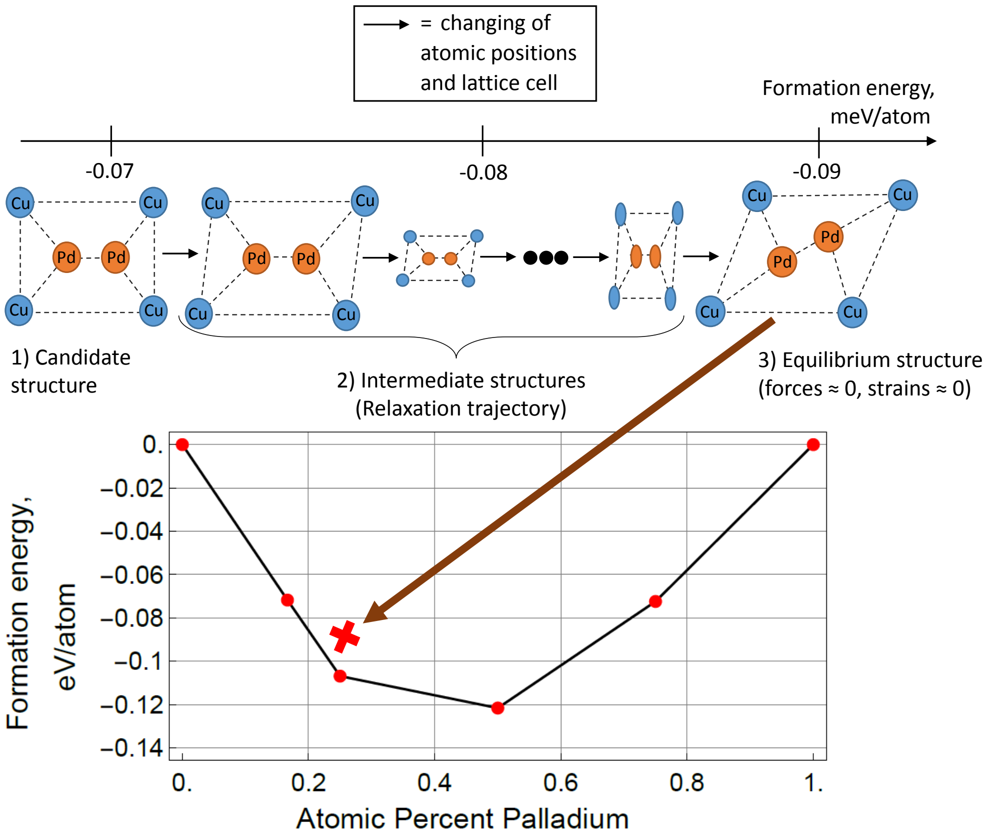

= changing of atomic positions and lattice cell
Formation energy, meV/atom
-0.07
-0.08
-0.09
Cu
Pd
1) Candidate structure
2) Intermediate structures (Relaxation trajectory)
3) Equilibrium structure (forces ≈ 0, strains ≈ 0)
Formation energy, eV/atom
0.
−0.02
−0.04
−0.06
−0.08
−0.1
−0.12
−0.14
0.
0.2
0.4
0.6
0.8
1.
Atomic Percent Palladium

Figure 11: Graphical illustration of the relaxation process. By *relaxation trajectory* we mean a sequential list of structures that occur during the relaxation, which have similar but distinct atomic displacements and lattice parameters. To perform relaxation we treat energy of configuration as a function of atomic positions and lattice vectors. As we know the derivatives of energy function w.r.t. this variables (calculated from forces and stresses), we use BFGS algorithm (as in Section 3.1) for minimizing energy, which simultaneously provides zero forces and stresses.

nature learn the short-range order by intrinsically assigning different energy contributions to different atomistic environments. This way, more energetically favourable short-range orderings will be distinguished among all possible orderings.

The active learning approach proposed in this section allows for "exploring" the unknown parts of the configurational space during a simulation: e.g., if an atomistic system during an MD is in the state preceding a phase transition, the system will evolve towards the new phase with increasing number of extrapolative configurations occurring. Whether the phase transition will be reproduced correctly depends on many factors, among which the accuracy of the MTP fitting is crucial. However in principle, atomistic simulations with active learning are designed to be suitable for such problems without the *a priori* knowledge about the studying system: the only source of information which "guides" the simulation is the QM values for energies, forces and stresses.

Due to the same reasons, the active learning approach can be applied to the simulations involving alloys with interstitial elements like C or H. The main difficulty here is: potential variety of local neighborhoods with H or C atoms (on which MTP should learn) is enormously huge to be represented in some generic training set, and most likely too unpredictable for being represented by a hand-made training set. Nevertheless, in active learning approach the MTP learns on relatively small amount of the representative atomic neighborhoods containing H and C, thus making such a simulation tangible from the computational point of view.

4 Results and Discussion

In this section the application of MTPs and the active learning approach to studying the properties of multicomponent systems is described. The two major parts of this section include CSP for binary and ternary alloys, and prediction of the properties of organic molecules.

4.1 Crystal Structure Prediction for Alloys

Advances in computer power, improvements in first-principles methods, and the generation of large materials databases like AFLOWLIB [16], OQMD [69], CMR [48], NOMAD [2], and Materials Project[42] have enabled modern data analysis tools to be applied in the field of materials discovery [44, 3, 62]. There have been growing efforts in computational search for materials with superior properties, including metallic alloys [59, 17, 34], semiconductor materials [37], and magnetic materials [70]. In this work we consider the problem of predicting stable phases in multicomponent alloys. A typical prediction algorithm consists of sampling structures across the configurational space and evaluating their energies. The sampling is done by searching through structures that are either selected from some carefully assembled pool of possible structures, often called crystal prototypes [54], or are generated by some sampling algorithm, see, e.g., Refs.[35, 52]

The evaluation of the energy of the structures in the pool is often done with density functional theory (DFT). Even despite its favorable accuracy/ efficiency trade-off as compared to other quantum-mechanical algorithms, the DFT calculations remain the bottleneck in materials prediction workflows, making an exhaustive search impractical. Machine learning (ML) for materials prediction has the potential to dramatically reduce the number of quantum-mechanical calculations performed and thus reduce the computational expense of predicting new materials via computation. The reduction of the computational time is achieved by constructing a surrogate model that "interpolates" the quantum-mechanical training data and makes subsequent energy evaluations much faster (by orders of magnitude). This is similar in spirit to the cluster expansion method which has been broadly used in different materials discovery applications [89, 43, 83, 37]. Cluster expansion is quite successful when the stable structures are derivatives of a particular structure (fcc, bcc, etc.) but is not useful when this is not the case. Its accuracy also converges slowly when atomic size mismatch is not negligible [58]. Additionally, more classical machine-learning algorithms such as decision trees [55], support vector machines [84], and other ML algorithms [75, 87] have been tried. However, in comparison to the standard machine learning approaches the moment tensor potentials-based [76] approach we demonstrate here provides broader applicability and can achieve higher accuracy as it is generic and apart from the charge and magnetism neglecting does not have any fundamental constraints on the fitted data, allowing for its effective application to different classes of materials.

The two important features of our approach are a completely general form for the interatomic potentials and an active learning algorithm for generating and refining the training set. In our approach, a ML model reproduces DFT for off-equilibrium structures that are not restricted to any lattice. Furthermore, the model learns the DFT interaction actively (on-the-fly) while equilibrating the candidate structures, completely automating the construction of the training set. Thus, structural optimization of the candidate structures can be performed via the interatomic potentials, rather than via DFT, further accelerating the construction of the training set.

Our method is based on moment tensor potentials (MTPs [76]) and the active learning algorithm [60]. Namely, we solve the following problem: given a set of elements, find the most stable structures (in the sense of lying on the convex hull of formation enthalpies) consisting of these elements, each characterized by composition, unit cell geometry and atomic positions within the unit cell. In this work we extend the interatomic potential [76] and active learning algorithm [60] to handle atomistic configurations with multiple types of atoms, similarly to the approach used in organic chemistry predictions [32].The differences between the algorithms from Ref.[32] and this work include that (1) we need derivatives of the energy, whereas in Ref.[32] we needed only the energy (or other predicted properties); and (2) that in Ref.[32] we were concerned with the selection from a finite set of predefined structures, whereas in this work we need to solve the problem predicting the energy with a fitted potential and assembling the training set used for the fitting at the same time (in other words, exploring the potential energy landscape and constructing the training set at the same time).

The idea of applying neural networks, as a broad class of machine-learning algorithms, to constructing interatomic potentials was pioneered in Ref.[10]. Application of Gaussian process regression, another class of machine-learning algorithms, was then proposed in Ref. [7]. The promising results obtained in these works have motivated many research groups to pursue this research direction [4, 8, 9, 11, 23, 27, 53, 57, 51, 77, 45, 81, 21, 22, 31, 82, 13, 49, 47, 15, 76, 72]. However, the application of such algorithms to the problem of materials prediction has proven difficult since following such a methodology requires one to collect all the representative structures in the training set which is as hard as predicting materials structure itself. In our view, it is the active learning [60, 61, 9, 12, 78] that paves the way for machine-learning interatomic potentials to accelerate computational materials discovery.

4.1.1 Cu-Pd Alloys

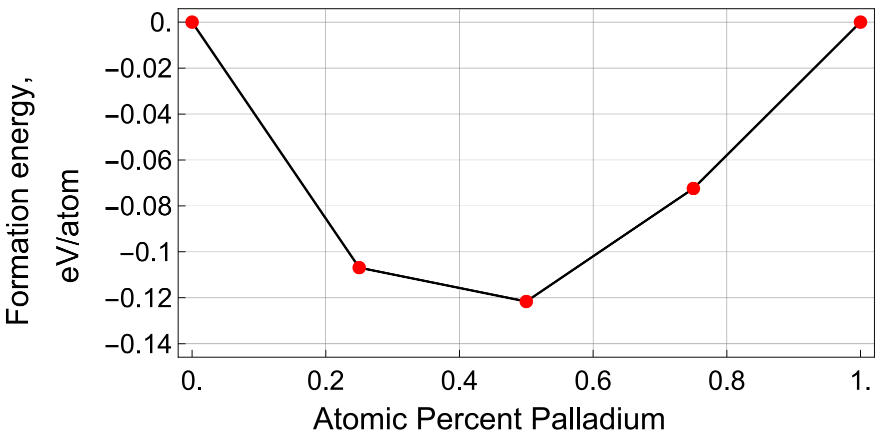
To test the applicability of our algorithm (Section 3.6.3) to the prediction of stable alloy structures we first used it to construct the Cu-Pd convex hull. We chose the Cu-Pd system because the structure of both pure Cu and Pd is fcc, while the stable equimolar CuPd structure is a bcc derivative structure. This system is a good test of whether or not our MTP-based model is able to simultaneously handle multiple lattice types.

We generated configurations with bcc, hcp, and fcc lattices with 12 or less atoms in the unit cell and populated them with Cu and Pd atoms in different combinations. This provided us with 40,000 candidate structures served as the input to our relaxation while learning-on-the-fly algorithm. We then equilibrated them and constructed a convex hull based on their relaxed energies. As follows from the scheme from Section 3.6.3, the training set increases on each iteration. The final training set was formed by 523 configurations. The energy MAE, RMSE (σ) and max. error measured on this training set were equal to 1.9 meV/atom, 2.3 meV/atom, and 10.1 meV/atom respectively. We call this training set “final” since an MTP trained on this set is able to relax all the candidate structures without exceeding threshold for the extrapolation grade. We used $\mathrm{lev}_{\max} = 16$ (refer to Section 3.4) to construct the MTP with about 200 parameters $\boldsymbol{\theta}$.

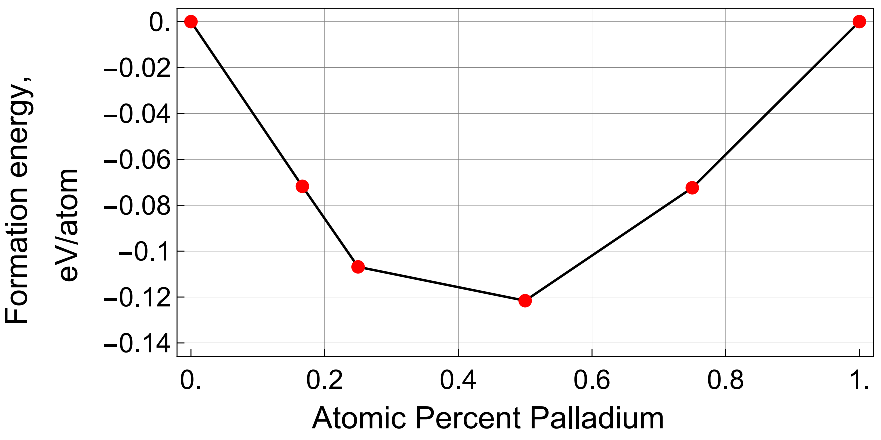
Figure 12 shows the convex hulls constructed by the MTP and by high-throughput DFT calculations as reported in AFLOW. To make a direct comparison possible, both convex hulls were post-relaxed with DFT using the same settings (such as pseudopotentials, k-point mesh, etc.). As a result, we have found a structure with 16.6% concentration of Pd that is not presented in the AFLOW library and has energy per atom 0.5 meV below the AFLOW convex hull level. Though such a shallow ground state (0.5 meV is comparable to thermal energy of one atom at 6K temperature, thus this minimum can easily be escaped due to thermal fluctuations in any realistic scenario) is typically not significant beyond academic interest, Cu-rich phases are believed to have an effect on the experimental Cu-Pd phase diagram and have been discussed in Refs.[5, 6] as a way of explaining the peculiar "off-stoichiometry" behavior on the Cu-rich side of the phase diagram.

It is illustrative to show the convex hull predicted by MTP and not post-relaxed with DFT. In Figure 13, only structures within the 4σ (10 meV/atom) interval from the MTP convex hull are shown. Visually, the MTP convex hull looks slightly different due to the approximation errors of MTP leading to different relative levels of the structures on the "energy per atom" axis. Still, MTP reproduced the stable phases present in AFLOW library.

During the entire procedure, most of the computational expense (about 90%) was DFT calculations. In total, we did 523 single-point DFT calculations on VASP. We used PAW_PBE GGA potentials, the energy cutoff was taken 500 eV, and K-mesh was generated automatically generated with a parameter KSPACING=0.15. If we relaxed all the 40,000 configurations using DFT, it would have taken about 10,000 times more computing time.



(a)



(b)

Figure 12: Comparison of the convex hulls (a) as obtained from AFLOW and re-calculated with DFT, and (b) as found by MTP and re-calculated with DFT. We have discovered a structure at 16.6% Pd which is 0.5 meV lower than AFLOW's convex hull.

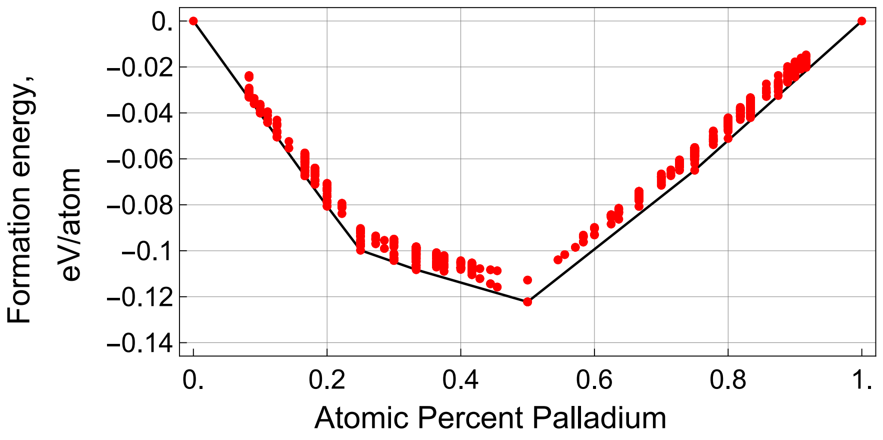


Figure 13: Convex hull constructed by MTP and structures with formation energy within 10 meV/atom above the convex hull.

4.1.2 Co-Nb-V Alloys

We next test our algorithm on constructing a convex hull for the ternary Co-Nb-V system in the region where the concentration of Co is 50% or more. The choice is motivated by the several Co-Nb and Co-V binaries present in this region of the phase diagram, which our approach should predict. The number of initial candidates was about 27,000 and they were bcc-like and close-packed (fcc, hcp, etc.) configurations with 8 or less atoms in the unit cell and different concentrations of Co, Nb and V.

The MTP was trained on-the-fly and the final training set consisted of 383 configurations with energy MAE, RMSE and max. error of MTP as 6.2 meV/atom, 8.1 meV/atom, and 29 meV/atom respectively. They were calculated with VASP DFT using PAW_PBE GGA potentials, the energy cutoff 400 eV, and K-mesh generating automatically with a parameter KSPACING=0.15. To consider magnetic properties of Co present in this alloy, we initialized calculations with parallel magnetic moments assigned to Co atoms, and zero magnetic moments assigned to Nb and V atoms, thus searching for a ferromagnetic ground state of a certain structure.

The resulting convex hull is shown in Figure 14. Remarkably, we have discovered a new structure with composition Co_3Nb_2V. It has a formation

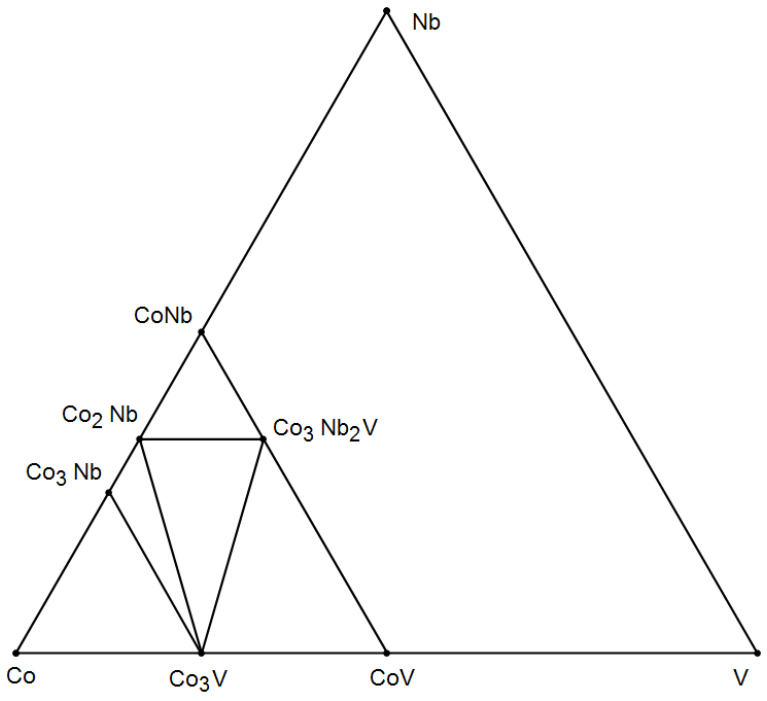


Figure 14: Convex hull of the Co-Nb-V system constructed by MTP in the Co-rich region.

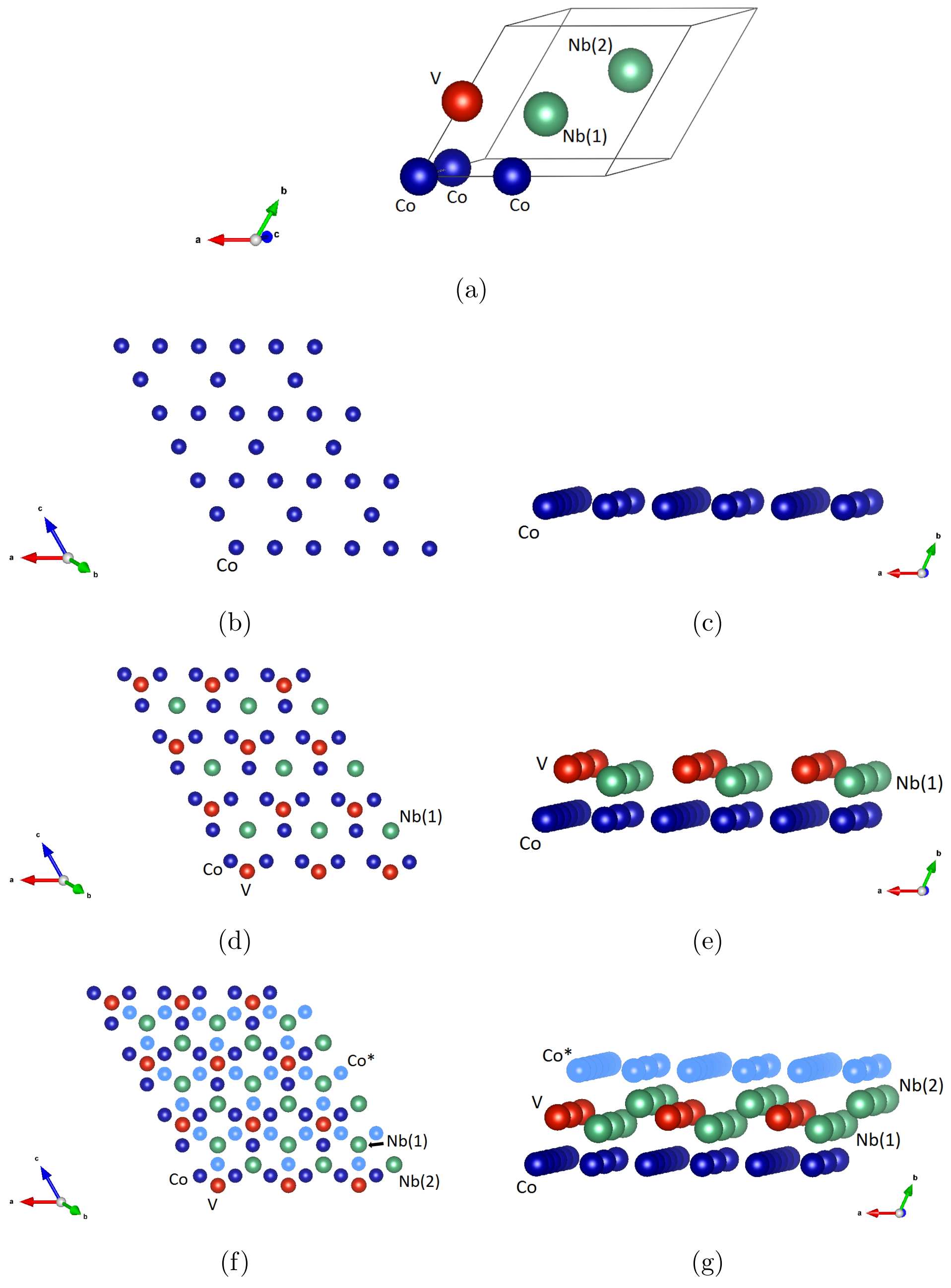


Figure 15: The Co_3Nb_2V discovered by MTP. The unit cell is shown in (a), while layer-by-layer plots in vertical and side projections are shown in (b)–(g). Co* show were the next (periodically extended) layer of Co atoms are positioned. The structure was found, although no similar crystal prototypes were used.

energy of 50 meV/atom below the AFLOW convex hull. Its unit cell and a layer-by-layer plot are shown in Figure 15. We remark that geometrically this structure is different from any of those in the initial pool—e.g., the Nb atoms have 16 nearest neighbors with distances between 2.76 and 2.98 Å. It would hence be impossible to accurately treat such a configuration for both an on-lattice model, such as cluster expansion, and an off-lattice model unless such a crystal prototype was known and explicitly added to the training set. This demonstrates the capabilities of our approach, combining an accurate off-lattice model and active learning.

4.1.3 Al-Ni-Ti Alloys

Finally, we applied our algorithm to the Al-Ni-Ti system. This system is well-studied and has many known ternary structures, some of which have over 20 atoms in the unit cell, therefore we considered this system a good test for our approach. We hence chose a set of candidate structures consisting of two parts: the first part has 1463 structures which were used in AFLOW as crystal prototypes.

The second part was generated for us by the authors of the algorithm from Ref.[35], which enumerates all possible unit cells with different symmetries (bcc, fcc and hcp) and different number of atoms; we have chosen unit cells containing up to 12 atoms, which results in 375,000 binary and ternary structures.

Including crystal structure prototypes adds extra difficulties: the structures may contain short interatomic distances (if, e.g., the original structure from which the prototype was derived had carbon-metal bonds which are shorter than typical metal-metal distances) and also smaller volume than that of the typical Al-Ni-Ti structures. Both of these features of the prototypes might result in unphysical structures with large stresses and forces which, in turn, lead to large MTP prediction errors. To make the unit cells of the candidate structures less deformed, we adjusted their volumes enforcing the relation:

$$v(n_{\mathrm{Al}}, n_{\mathrm{Ni}}, n_{\mathrm{Ti}}) = n_{\mathrm{Al}} v_{\mathrm{Al}} + n_{\mathrm{Ni}} v_{\mathrm{Ni}} + n_{\mathrm{Ti}} v_{\mathrm{Ti}}, \tag{30}$$

where $v(n_{\mathrm{Al}}, n_{\mathrm{Ni}}, n_{\mathrm{Ti}})$ is the volume per atom assigned to the unit cell with concentrations of Al, Ni, Ti equal to n_{Al}, n_{Ni}, n_{Ti} respectively and v_{Al}, v_{Ni}, v_{Ti} are the volumes per atom for equilibrium fcc-Al, fcc-Ni, hcp-Ti structures respectively. Resizing the unit cells in this way provides an initial guess for their volumes (a kind of Vegard's law for different lattice types.)

To circumvent the large prediction errors that might occur for prototype structures with bond lengths and neighborhoods atypical of alloys, we performed a two-step relaxation as explained below. We used $\mathrm{lev}_{\max} = 20$ (see Section 3.4) to construct the MTP with about 650 parameters. This makes the potential more accurate, but requires more data for training, than with $\mathrm{lev}_{\max} = 16$. First, we did the same procedure as for the Cu-Pd and Co-Nb-V systems, which provided us with the training set of 2393 configurations with *ab initio* energies, forces and stresses. They were calculated with VASP DFT using PAW_PBE GGA potentials with energy cutoff 400 eV, and K-mesh generated automatically with a parameter KSPACING=0.15. To consider magnetic properties of this alloy, we initialized calculations with parallel magnetic moments assigned to Ni atoms, and zero magnetic moments assigned to Al and Ti atoms, thus searching for a ferromagnetic ground state of a certain structure. The MTP trained on this set has energy MAE, RMSE and max. error as 18 meV/atom, 27 meV/atom, and 91 meV/atom respectively.

We next relaxed the 377,000 configurations and constructed a convex hull. Next, we picked all the configurations whose formation energy per atom is lower than 4σ ($\approx$ 100 meV) from the convex hull level. This left us with 62,000 configurations.

Second, we repeated the procedure of relaxing the 62,000 configurations on-the-fly from scratch, starting from an empty training set. During this process a new training set with 976 structures was formed by the active learning algorithm. The MAE, RMSE, and max.errors on this training set were 7 meV/atom, 9 meV/atom, and 24 meV/atom respectively.

This way we constructed a convex hull based on more accurate formation energies, than would be possible after the first step.

Formula	Energy relative to the convex hull
Al_4Ni_8	-7.38 meV
Al_1Ni_{11}	-1.18 meV
$Al_1Ni_9Ti_2$	-0.34 meV

Table 1: New Al-Ni-Ti structures found in this study. The "level below the convex hull" was computed using DFT.

To perform a comparison with the AFLOW convex hull, from the 62,000 relaxed configurations we eliminated all the configurations with formation energy per atom higher than 4σ from the convex hull level, where now $\sigma = 9$ meV/atom. This left us with about 7000 configurations, which were subsequently relaxed with DFT. After this, we constructed a final convex hull using the DFT formation energies. It has all the structures, present in AFLOW, and three new structures discovered by MTP (see Figure 16). Their chemical formulas are given in Table 1 together with their position below the AFLOW convex hull level. Interestingly, all the structures are Ni-rich which makes their discovery relevant to the application of Ni-based alloys.

Taking into account that after the first step we have obtained an MTP capable of relaxing all the 377,000 configurations, we call it the "robust" potential. After the second step we have obtained an MTP which is trained on (and thus able to relax) the low-energy near-equilibrium structures only. We refer to this MTP as the "accurate" potential. We attribute the difference in accuracies of the robust and accurate potentials to the fact that, at the second step, the trajectories of relaxations started from near-equilibrium structures (within the accuracy of the robust potential), see an illustration in Figure 17. This reduces the region in the configurational space in which the MTP is fitted, thus improving the accuracy in comparison to the first step.

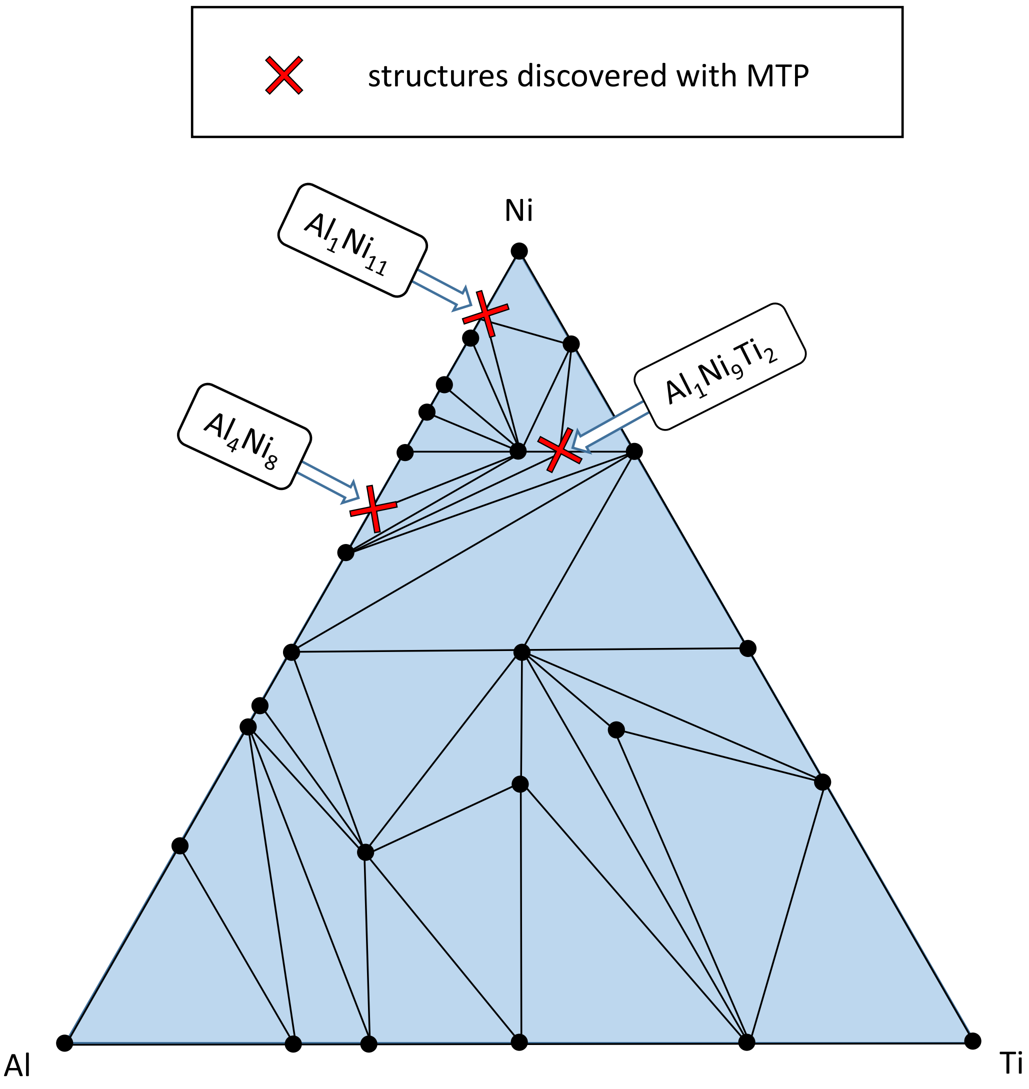


Figure 16: Al-Ni-Ti convex hull constructed by MTP and compared to the one from AFLOW. The MTP convex hull contains all the structures from AFLOW plus three newly discovered ones.

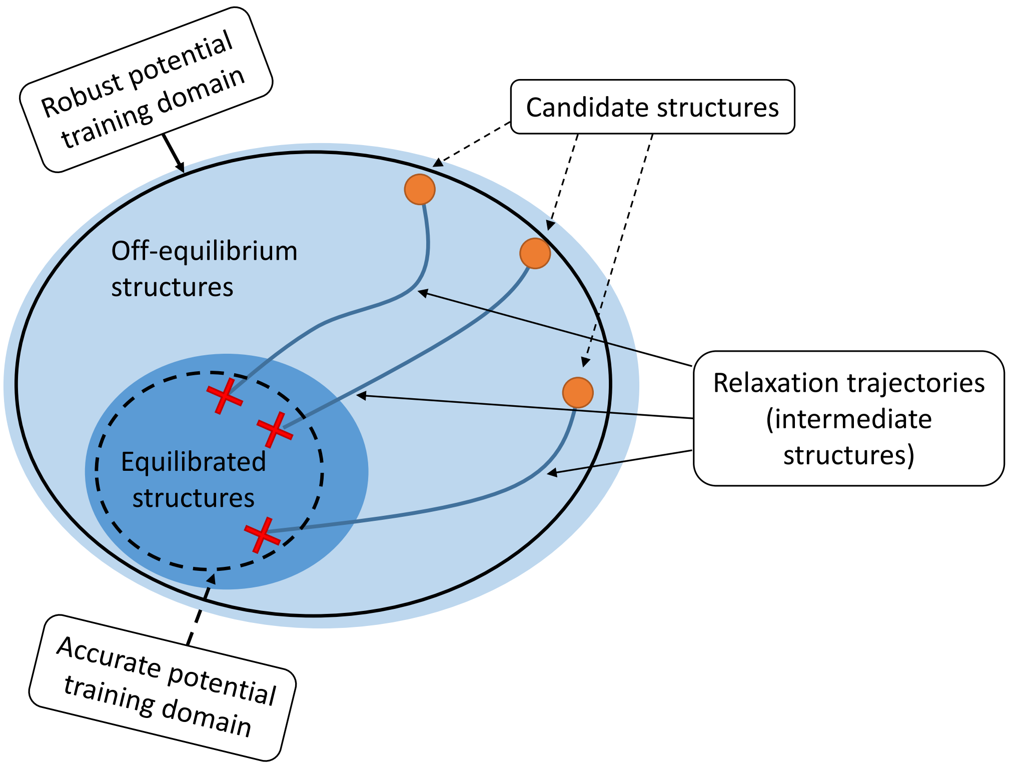

Robust potential training domain
Candidate structures
Off-equilibrium structures
Relaxation trajectories (intermediate structures)
Equilibrated structures
Accurate potential training domain

Figure 17: The accurate potential is trained on a smaller domain of configurational space than the robust one. Thus, the accurate potential provides more accurate predictions at the interior of the “Relaxed structures” region.

4.1.4 Discussion

We have developed an algorithm for constructing a convex hull of stable alloy structures based on the moment tensor potentials (MTPs) to approximate *ab initio* energies, forces and stresses of atomistic configurations. This way the calculations for atomistic systems can be done much faster than with DFT, while the accuracy is comparable to that of DFT. The active-learning algorithm forms a training set automatically, removing the need for its manual design—the most tedious part of application of ML to atomistic modeling. We have verified the applicability of our algorithm by constructing the convex hulls for the Cu-Pd, Co-Nb-V and Al-Ni-Ti metallic alloy systems and comparing them to the convex hulls from AFLOW library. For all the systems we have discovered new stable structures, which are not listed in the AFLOW library. We attribute this to the large amount of candidate structures (40,000 for Cu-Pd, 27,000 for Co-Nb-V, 377,000 for Al-Ni-Ti) we explored, which would be impossible to equilibrate using DFT. Instead, we performed relaxations using fast MTP calculations, referring to DFT only for the training data generation. In the cases covered by this paper, the amount of single-point DFT calculations was about 1% of the total amount of relaxed configurations. In comparison to the high-throughput DFT calculations, the speedup is three to four orders of magnitude.

4.2 Predicting the Properties of Organic Molecules

A permanent demand for the discovery of new compounds in numerous fields of industry requires development of the computational tools for prediction of molecular properties. There are many quantum-mechanical algorithms that are able to accurately predict properties of, theoretically, arbitrary atomic systems, however in practice these algorithms are too computationally expensive to be applied to a very large number of molecules. Density functional theory, which is frequently used due to its favorable trade-off between accuracy and computational cost, is still too time-consuming for high-throughput (rapid) screening of a large number of molecules.

In this work we propose a new algorithm of fitting of molecular properties. Our model resembles neural networks in the sense of employing several computing layers. We show that our model requires less training data to achieve the chemical accuracy when compared against the state-of-the-art approaches on the existing benchmark tests. For example, on the benchmark database consisting of 130k molecules [63] the majority of recent state-of-the-art algorithms achieve chemical accuracy only when trained on tens of thousands of samples, while our model does it with only few thousands of samples. We attribute this to our local model of interatomic interaction that effectively relates the molecular properties to the atomic environments and makes predictions for the molecules not present in the training set by accounting for contributions of the individual atomic environments.

The other problem we address with our algorithm is the issue of identifying the so-called *outliers*. In the discovery of new molecules the most "interesting" molecules are often the most atypical ones. This is a challenge for ML approaches: if no molecules with similar structure are present in the training set, ML models extrapolate and commit large prediction errors for these outliers. In the proposed active learning algorithm we have a criterion for detecting the outliers (even if their *ab initio* properties are unknown, see Section 3.6.2), reducing the errors by including the molecules with the most different geometries and compositions in the training set. This prevents the cases when we are trying to predict properties of molecules that are too different from any of the training samples; instead the properties of the molecules outside the training set are interpolated by the ML model and are thus predicted accurately.

This paper has the following structure: we first formulate the problem of prediction of molecular properties in an ML framework and present our machine-learning model as a solution. Then, in Section 3.6 we describe our active learning algorithm. In Section 4 we compare our algorithm to the existing algorithms [14, 33, 63, 64, 65, 39, 73, 24, 29, 41] on the two benchmark datasets: QM9 [63] and QM7 [33]. The concluding remarks are given in Section 4.2.6.

We have conducted a number of tests in order to clarify the following three questions: what accuracy can our model achieve on a dataset when trained on its randomly chosen subset, how the accuracy can be improved using our active learning technique (Section 3.6), and how many training samples are required to reach the chemical accuracy, 1 kcal/mol. We remind that we use the following terminology: the *training set* is the set on which we train our model and the *validation set* consists of the full database excluding the training set. All errors quoted below are measured on the validation set.

We have tested the models of level 16, 20 and 24 denoted by MTM_{16}, MTM_{20}, MTM_{24}. The fitting of the models was done with the BFGS optimization algorithm, by performing between 2000 and 5000 iterations.

4.2.1 Fitting Enthalpy on QM9

First, we fit our model on the so-called QM9 dataset [63] consisting of 130831 molecules formed by C,H,O,N,F atoms with up to nine heavy (C,O,N,F) atoms. This is a subset of the originally proposed database [66] consisting of 166 billion of organic molecules. The number 130831 excludes 3054 molecules from the database that failed a consistency test, as reported in Ref.[63]. Following the existing works [64, 65, 73, 29, 41] we demonstrate the performance of our method by fitting the enthalpy (or atomization energy) at 300 K.

4.2.2 Random Choice of the Training Dataset

To investigate the accuracy of the MTMs with different number of parameters we have calculated the learning curves showing the dependence of the mean absolute error (MAE) on the training set size, see Figure 18. Our results are averaged over three independent random choices of the training set. As expected, the models with fewer parameters show good results for small training datasets, but are outperformed by the models with more parameters as the number of training samples grows.

We next compare different models by how fast (i.e., with what training

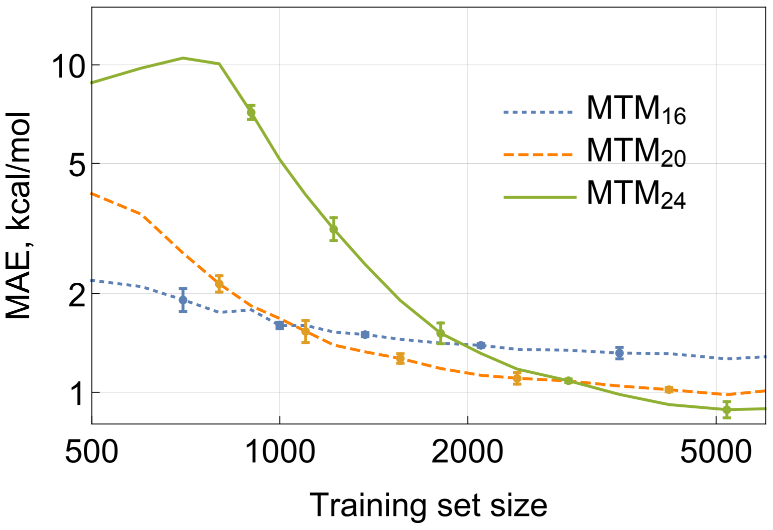


Figure 18: Random selection of the training set: dependence of MAE on the training set size. Different curves show different models: the higher is the number, the more parameters the model has.

dataset size) they reach chemical accuracy. Table 2 lists the prediction errors for the MTMs and the existing state-of-the-art methods when training on random training sets. While filling this table, we used MTM_{16} for the 1k training set size, MTM_{20} for the 3.5k training set size, and MTM_{24} for the 10k training set size. For the sizes of 25k and 50k we used the MTM_{28} model which has more parameters to fit then MTM_{24}. At the same time, from Fig. 18 it can be seen that using either MTM_{20} or MTM_{24} models still provides competitive results.

The aSLATM [40] model reaches the chemical accuracy with the training set size about 3200, while the learning curve from Figure 18 shows that MTM_{24} model requires almost the same number, 3500 molecules to reach such accuracy. We obtained the results for MTM_{16}, MTM_{20}, and MTM_{24} by doing 2000 iterations of the BFGS algorithm and 5000 iterations for MTM_{28}. For a more accurate comparison with aSLATM, we trained MTM_{24} model on 10 different random samples of training set with 3000 molecules by doing 3000 iterations of BFGS. This provided us with a validation MAE of 1.006 ± 0.0316

	Training set size						
Model	1k	3.5k	10k	25k	35k*	50k	110k
DTNN[73]	-	-	1.2	1.0	-	0.94	-
BAML[39]	-	-	2.4	-	-	-	-
$\Delta_{MP7}^{B3LYP} - ML$[65]	4.8	-	3.0	-	-	-	-
MPNN[29]	-	-	-	-	-	-	0.39
HDAD[24]	-	-	-	-	1.0	-	0.58*
HIP-NN[50]	-	-	-	-	-	**0.35**	**0.26**
SchNet[71]	-	-	-	-	-	0.59	0.31
aSLATM[40]	**1.8**	**0.98**"	-	-	-	-	-
MTM†	**1.8**	**1.0**+	**0.86**	**0.63**	-	0.41	-

Table 2: Comparison of the MAE of prediction of atomization energy (kcal/mol) by different models for different training set sizes.

† *this work*

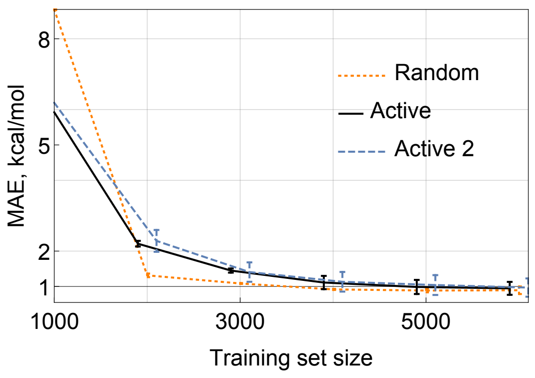
**As estimated from the graphs in Ref.[24].*

+*The error can be decreased by further training, see the text.*

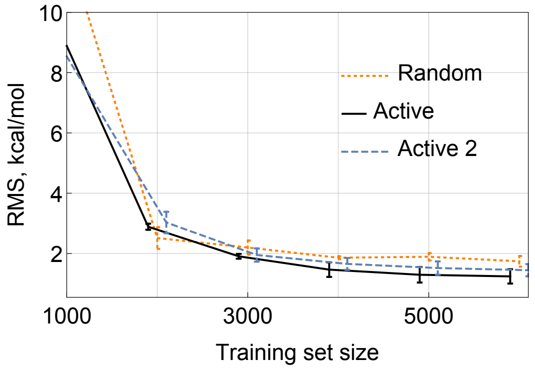
" *as estimated from the graph in Ref.[40]*

kcal/mol.

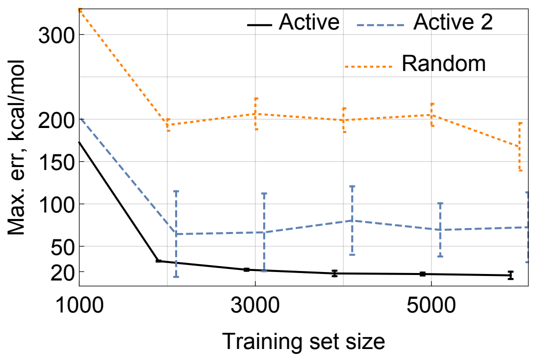
As can be seen from Table 2, for training set sizes not more than 10k MTM shows a better learning curve than the existing state-of-the-art methods and only the model from the Ref.[40] shows nearly equal results. The works using deep NNs (Refs.[71] and[50]) show better accuracy for training set sizes over 50k. Ref.[73] and[24] need 25k and 35k training samples to attain the chemical accuracy, respectively. We note of another work [29] that also reaches the chemical accuracy, but it only reports MAE of 0.55 kcal/mol while training on 110k molecules plus another 10k molecules used as a hold-out set for the early stopping criterion.



(a)



(b)



(c)

Figure 19: Active and random selections of molecules: dependencies of MAE (a), RMSE (b) and maximal error (c) on the training set size for the MTM_{24} model. Active and Active 2 lines stand for scenarios, when MTM can and can not pick configurations from the set on which the errors are measured.

4.2.3 Active Learning

While the MAE of MTMs trained on random training sets is small, the corresponding maximal error is of the order of 100 kcal/mol, resulting in the outliers for which the atomization energy prediction is too inaccurate. To get rid of the outliers we have applied the active learning algorithm as described in Section 3.6. We started with a random training set of 1k molecules and it takes about 20 iterations to reach the training set size of 6k molecules. At each iteration, our training procedure takes an amount of time proportional to the training set size, while the selection always takes nearly constant time, approximately the same as required to train a model on a training set with 1k molecules.

Figure 19 shows the graphs of the MAE, RMSE and maximal absolute errors depending on the training set size, comparing random and active selection. The "Active" curve corresponds to the scenario of actively selecting molecules from the entire set of molecules and measuring the error on the remaining set. In the "Active 2" scenario we instead separate out a validation set of 30k molecules on which we measure error, while selecting and learning from the remaining 100k molecules. The error bars correspond to the 95% confidence interval as measured on three independent runs, in each of which the initial training set of 1000 molecules and the validation set in the "Active 2" scenario were random. We have used the MTM_{24} model. It has about 2000 parameters to fit, which explains why the error on the Figure 19 exhibits *overfitting* when trained on less than 2000 molecules.

In the active learning approach (see Section 3.6), the training set tends to be as diverse as possible in the sense of spanning the largest volume of configurational (or molecular) space. The most extrapolative molecules lie on the boundaries of this volume and if the amount of molecules is less than or close to the number of model parameters we would not have sufficient amount of training samples in-between the boundaries. With the random selection we cover the configurational space more evenly, which results in lower RMSE. From the part of the graph starting from 4000 training samples (this is twice the number of model parameters, as we suggested as a rule of thumb in Section 3.1) we can see that there is no improvement in MAE (though it is rather close to the random-sampling MAE), but the RMS and maximal errors are lower that for the random sampling.

From Figure 19 it can be seen that the maximal error is much less when we are allowed to add any possible molecule to the training set (the “Active” scenario) as compared to the “Active 2” scenario, however, the error in the “Active 2” scenario is still smaller than that in random sampling. This indicates that the active learning have two mechanisms of decreasing the error: the actively chosen molecules represent better the unusual molecules in the validation set of the “Active 2” scenario, but if we are allowed to select also those unusual molecules for training, the error further drops. We argue that the latter (“Active” scenario) can be useful in those applications where the region of interest in the chemical space is fixed *a priori*. As an example of such application, in [[59]] the authors found six candidates of Co superalloys from about 2k *a priori* identified potential chemical compositions.

To better understand the impact of active learning, on Figure 21 we plotted the true and predicted enthalpies for the models trained on randomly and actively chosen training sets of 10k molecules. The plot is focused on a small region of enthalpies to show the scale of the error. As can be seen, active learning makes the error small uniformly over all the samples, while the random choice of the training set results in outliers for which the error is large.

We investigated the sizes of the molecules entering the training set at each iteration of selection (see Section 3.6.2) and compared them to the average molecule size among the QM9 database. The results are shown on Figure 20. From this graphs it can be seen that the algorithm tries to select molecules with sizes lower than average. This could indicate that some small molecules contain representative atomic neighborhoods which occur in a many bigger molecules and the active learning algorithm detects and selects such small molecules. This result is in correspondence with the observations of Huang

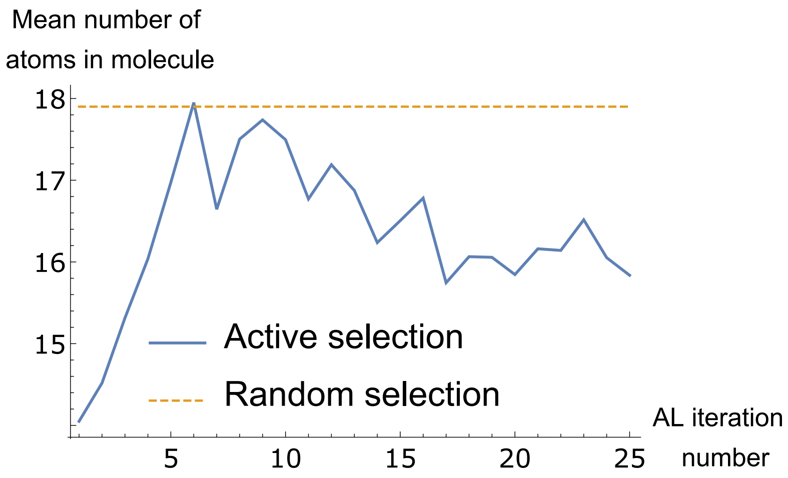
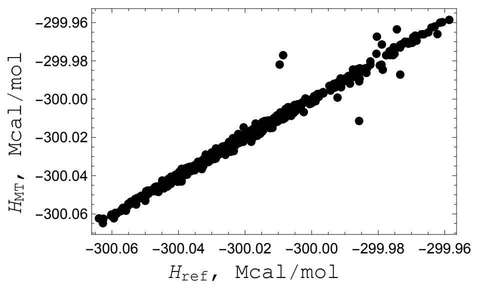


Figure 20: Mean numbers of atoms in a molecule for active and random selections of configurations. The MTM_{24} was used for this experiment.

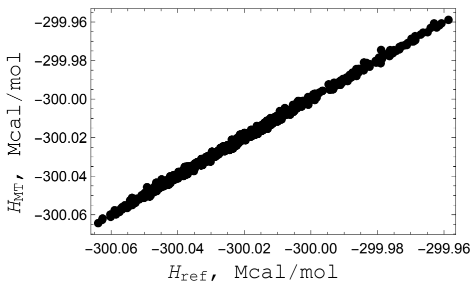
et al. [40], where the authors state that an accurate model can be obtained by training on a small amount of the most relevant atomic environments (neighborhoods).

4.2.4 Fitting the QM7 Database

Another common benchmark database we used consists of 7.2k small organic molecules with up to seven heavy atoms (C,N,O,S) saturated by H, and is referred to as QM7 [56]. We have chosen the four properties to fit: atomization energy, polarizability, and HOMO and LUMO levels. The atomization energy and polarizability are extensive quantities, and the other two are intensive quantities. The training set consisted of 5k randomly chosen molecules while the remaining 2200 were used for validation. In Table 3 we compare the accuracy of the MTMs to the other state-of-the-art methods. Our results are averaged over three independent runs. We see that the local properties, enthalpy and polarizability, are fitted with the same or higher accuracy as by the existing methods, while predictions of the HOMO (highest occupied molecular orbital) and LUMO (lowest unoccupied molecular orbital) levels have a 50% larger error than the state of the art.



(a) random choice of training samples



(b) active choice of training samples

Figure 21: Scatter plot of the formation enthalpy predicted by machine learning, H_{MT}, versus the reference enthalpy H_{ref}, with (a) random and (b) active choice of training samples. For the illustration purposes we have plotted a small area from the whole database, where the errors are significant.

To address this issue, we have applied the nonlocal modification of our algorithm, nlMTM, as given by (32). From Table 3 we see that accounting for nonlocality improves the error of HOMO and LUMO to the state-of-the-art accuracy. As, the nonlocal scheme is, essentially, a three-layer model, it was found to suffer from overfitting similarly to the deep neural networks; therefore we used the early stopping technique for training the nlMTM. To that end, we used 1100 samples for estimating the error during training and measured the error on the remaining 1100 samples.

4.2.5 A Nonlocal Model

MTP, being a local model (22), can successfully describe interatomic interactions when they are effectively screened. This holds true for a lot of cases, however some molecular quantities may be essentially determined by multiatom interactions which can span distances far longer than cut-off radius.

Taking into account relatively poor performance of conventional MTMs for HOMO and LUMO molecular quantities (see 3) we decided to include nonlocal effects by introducing two different local models V_1 and V_2 (each

Model	E (kcal/mol)	α (Å^3)	HOMO (eV)	LUMO (eV)
BAML[39]	1.15	0.07	**0.10**	**0.11**
SOAP[19]	0.92	0.05	0.12	0.12
DTNN[73]	1.04	-	-	-
MBTR[41]	0.60	**0.04**	-	-
MTM†	**0.52**	**0.04**	0.16	0.16
nlMTM†	-	-	0.11	**0.11**

Table 3: Mean average errors of predicting the atomization energy E, polarizability α, and HOMO and LUMO levels, committed by different models on the QM7 dataset.
† *this work*

with its own set of parameters) and let

$$\begin{aligned} v_1 &= \sum_i V_1(\mathfrak{n}_i), \qquad \text{and} \\ v_2 &= \sum_i V_2(\mathfrak{n}_i). \end{aligned} \tag{31}$$

We then define the nonlocal model *nlMTM* in the following form

$$\begin{aligned} F_{\rm nl}(x) = {} & p_1 v_1 + p_2 v_2 + p_3 v_1^2 + p_4 v_2^2 + p_5 v_1 v_2 \\ & + p_6 v_1^3 + p_7 v_2^3 + p_8 v_1^2 v_2 + p_9 v_1 v_2^2. \end{aligned} \tag{32}$$

Figure 22 shows how the parameters p_i form the additional computing "layer", if compared to the local model in Figure 1 and extend the set of parameters, $\boldsymbol{\theta} = \big(\xi_\alpha, c^{(k)}_{\mu,z_i,z_j}, p_i\big)$, that are found in the training process. Such a model architecture is motivated by the fact that molecular orbitals depend largely on local environments described by several V_i (although we only considered $i = 1, 2$). However, these orbitals get occupied by electrons not independently of each other, hence we assume a nonlinear dependence of the answer on the local features (32).

E.g., on Figure 23 the neighborhoods of atoms 1 and 4 do not intersect, therefore neither pair interaction 1-4 nor triple interactions 1-2-4 or 1-3-4, etc.

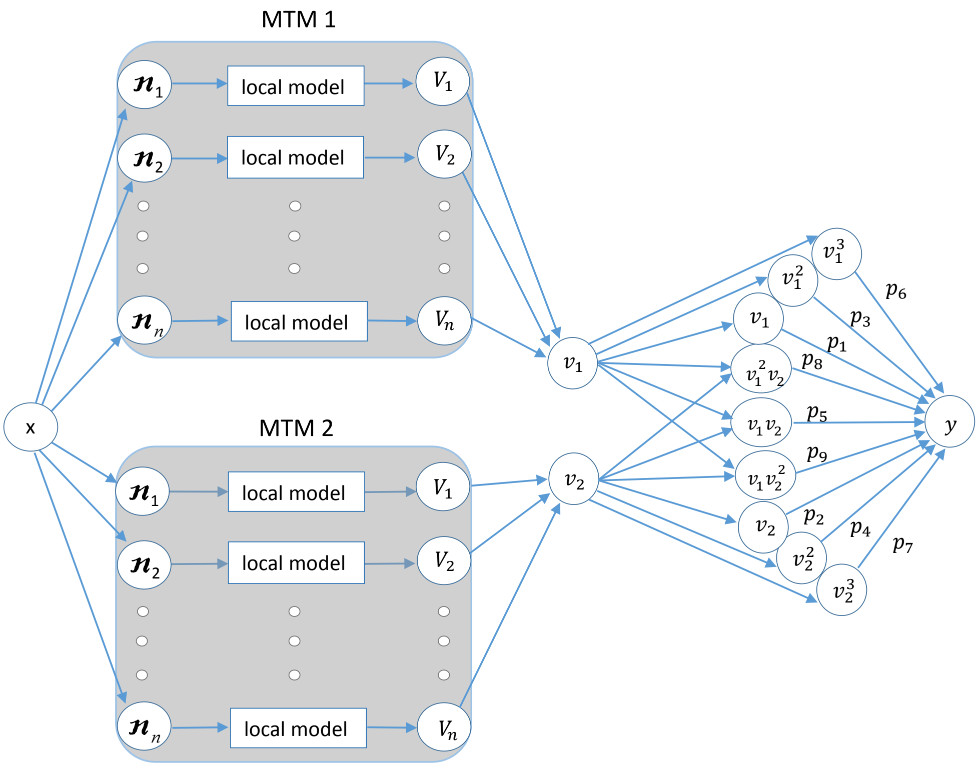


Figure 22: The nonlocal model, nlMTM, constructed from the two local models. $p_1, \dots, p_9$ are the additional fitting parameters.

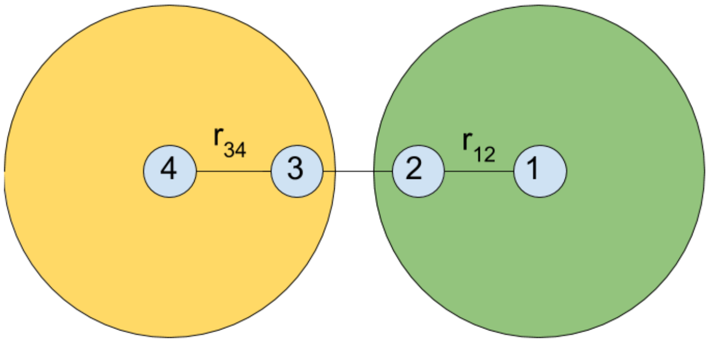


Figure 23: Neighborhoods of atoms 1 and 4 do not intersect. Therefore mutual interaction of atoms 1 and 4 cannot be taken into account by a local model with current cutoff radius.

cannot be approximated by the local MTM model. As opposite, following the formulas (32) and (31), the approximated nonlocal quantity will include, e.g., the following terms:

$$\begin{aligned} F_{\mathrm{nl}}(x) &= p_5 v_1(r_{12}, r_{34}, ...) v_2(r_{12}, r_{34}, ...) + ... \\ &= p_5 (V_1(r_{12}) + ...) * (V_2(r_{34}) + ...) + ... \quad = \underbrace{p_5 V_1(r_{12}) * V_2(r_{34})}_{f(r_{12}, r_{34})} + ... \end{aligned} \tag{33}$$

The underbraced summand in (33) depends on both r_{12} and r_{34}. Similarly, after expanding all V_1's and V_2's in (32) the expression for $F_{\mathrm{nl}}(x)$ contains polynomials of r_{ij} mutually multiplied independently of the distance between the corresponding atoms and thus forms a basis for approximating nonlocal properties similarly to local model, see Section 3.4.

4.2.6 Discussion

We have introduced the moment tensor model (MTM) for prediction of the molecular properties based on the contributions of local atomic environments represented by the moment tensor descriptors. The accuracy of the proposed model is comparable or better than that of the state-of-the-art algorithms. The proposed model outperforms the existing algorithms by how fast it reaches the chemical accuracy on the database of 130k molecules. We attribute this to the provable completeness of our moment tensor descriptors of local environments. It should be emphasized that although our descriptors (25) are, essentially, two-body descriptors, their completeness and, in particular, angular dependence comes from their tensorial structure.

In addition, we have proposed an active learning algorithm that significantly reduces the maximal error (in other words, the error for the outliers). The algorithm effectively selects the molecules most different from those already in the training set and adds these molecules to the training set.

As MTM allows calculating forces due its differentiable analytical form (Section 3.3) it is in principle capable of describing chemical reactions. However, this have not been explicitly tested for any particular reaction.

5 MLIP Software

The moment tensor potentials (see Section 3.4) and the active learning framework (see Section 3.6), which are used in a present research, are implemented in a MLIP software package. It contains tools related to working with machine-learning interatomic potentials and atomistic simulations and is primarily implemented in C++, with the Python version (the `mlippy` package) available as well. The repository is available via the following link:
https://gitlab.com/ashapeev/mlip-v2

As this software is in active development, its actual description may be outdated, but mostly in terms of the commands syntax, while the working principles remain the same. You can always find the actual examples in the repository itself.

5.1 Installation

5.2 MLIP Commands

In this section the features related to the multicomponent MTPs are discussed. One of the ways of using MLIP is executing commands with a compiled MLIP binary file. Launching a command with two options can be done in Linux shell by typing the text:

`./mlp command --option1=value1 --option1=value2`, where `command` is the name of the command and expressions of type `--option1=val1` are options with the corresponding values. To execute a command in parallel mode, type the
`mpiexec -n N ./mlp ...` or the
`mpirun -n N ./mlp ...` depending on your system, where `N` is the desired number of cores. In Table 4 the most common commands are listed.

Command	Description	Parallel
train	Fitting an MTP to a database	yes
calc-errors	Calculating the errors on a database	yes
select-add	Using active learning to add new configurations to the training set	yes
relax	Performing structural relaxation of given configurations using the MTP	yes
convert-cfg	Converting configurations between different formats	no
self-test	Launching MLIP self-testing utility	yes
list	Listing available MLIP commands	no
help	Showing info about a command	no

Table 4: List of the most common commands present in MLIP software.

5.2.1 The "train" Command

- Description

  Fitting an MTP to the database. This launches the training procedure (see Section 3.1), which results in finding the optimal parameters for a given MTP, in a sense of minimizing a certain functional, most closely related to the weighted sum of squared deviations of certain characteristics. This requires configurations from database to have at least one of the following quantities: energy, forces acting on atoms or stresses acting on lattice. Depending on the `weighting` option, the loss functional is composed in different ways, e.g., aimed at fitting energies or energies per atom best, which is achieved by dividing correspondent terms in equation (20) by the number of atoms in each configuration.

- Input

  pot.mtp - the file with an MTP to be trained. This can be both clean potential with uninitialized parameter values or a previously trained potential.
  train.cfg - the file with configurations in a `cfg` format to be used as training set.

- Output

  Trained.mtp - the file with a trained MTP.
  state.mvs - the file containing MaxVol state (see Section 3.6) of the trained MTP.
  Console: the options which are used in the current training procedure are printed. As well the values of the loss functional on each step of optimization algorithm (see Section 3.1) are printed. When optimization procedure ends, the training errors are printed in the same style, as in `calc-errors` command, see below.

- Options

  –energy-weight=*double*: weight of energies in the fitting. Default=1.
  –force-weight=*double*: weight of forces in the fitting. Default=0.01.

–stress-weight=*double*: weight of stresses in the fitting. Default=0.001.
–scale-by-force=*double*: If >0 then configurations near equilibrium (with force ≈ 0) get more weight. Default=0.
–valid-cfgs=*string*: filename with configuration to validate.
–max-iter=*int*: maximal number of iterations. Default=1000.
–trained-pot-name=*string*: filename for trained potential. Default=Trained.mtp.
–bfgs-conv-tol=*double*: stopping criterion for optimization. Default=10^{-8}.
–weighting=*string*: way of weighting the functional for better fitting of properties. Default=vibrations. Others=molecules, structures.
–init-params=*string*: how to initialize parameters if the potential was not pre-fitted. Default is "random". Other is "same"—the parameters are set with a certain default values.
–skip-preinit: skip the 75 iterations done when params are not given.
–auto-mindist: automatically decrease the min-dist in the MTP based on the training set. Default=false. Other=true or 1.
–mvs-filename: filename of the `mvs` file (containing the state of the MaxVol) to be created. Default=state.mvs. It is required by commands using active learning, e.g., the `relax` command.

5.2.2 The "calc-errors" Command

- Description
  Calculating the approximation errors of a given MTP on a given database. This requires configurations from database to have at least one of the following quantities: energy, forces acting on atoms or stresses acting on lattice. For each of such configurations MTP predicts these quantities based on its parameters values and compares to their reference values, listed in the database.

- Input
  pot.mtp - the file with an MTP. This should be a previously trained potential, otherwise the command makes no sense.

database.cfg - the file with configurations in a `cfg` format to be used for the errors checking.

- Output

  Console: the errors are listed for such quantities as energy, energy per atom, forces, stresses and virial stresses (stresses divided by volume of the unit cell). For each of these quantities the mean average errors, root-mean square errors and maximal errors are printed.

- Options

  –only-energy: calculate only energy (forces/stresses info should be ignored). By default this option is disabled.

5.2.3 The "select-add" Command

- Description

  Performing the selection of configurations, which should enter the training set, according to MaxVol criterion in active learning framework (see Section 3.6). Note, that actually the selection is performed not over configurations, but over equations, corresponding to certain configurations. The equations can be composed in different ways (see the description of `train` command), and thus the MaxVol grade of each configuration can be defined in different ways. Usually the selection just over the energies (as by default) is the best choice.

- Input

  pot.mtp - the file with an MTP. This should be a previously trained potential.
  train.cfg - the file with configurations in a `cfg` format containing a training set to which the MTP was fitted.
  new.cfg - the file with configurations in a `cfg` format containing a list of configurations, from which the selection is performed.
  diff.cfg - the filename with configurations to be added to the training set.

- Output

  diff.cfg - the file containing selected configurations.

  Console: The amount of selected configurations is printed. In case of parallel run, the amounts of selected configurations are printed for each process.

- Options

  –select-threshold=*double*: set the select threshold to num, default=1.1.
  –energy-weight=*double*: set the weight for energy equation, default=1.
  –force-weight=*double*: set the weight for force equations, default=0.
  –stress-weight=*double*: set the weight for stress equations, default=0.
  –nbh-weight=*double*: set the weight for site energy equations, default=0.
  –mvs-filename=*string*: save MaxVol state to this file.
  –selected-filename=*string*: file with selected configurations from both old and new training sets.
  –selection-limit=*int* limit the number of configurations to be selected, default=0 (no limit).
  –weighting=*string*: way of weighting the functional for better fitting of properties. Default=vibrations. Others=molecules, structures.

5.2.4 The "relax" Command

- Description

  Performing a structural relaxation, i.e., energy minimization of configurations using the MTP as a force-field. The relaxation is performed within the active learning framework (see Figure 9).

- Input

  settings.ini - the file of a special format, containing various settings of the relaxation.

- Output

  Console: the outcome of relaxation for each configuration is printed.

It can converge successfully, diverge (e.g., due to weakly trained potential), or can be terminated due to the high extrapolation grade (see Figure 9).

- Options

  –cfg-filename=*string* - the file with configurations, which should be relaxed.

  –save-relaxed=*string* - the file containing successfully relaxed configurations.

  –save-unrelaxed=*string* - the file containing configurations, whose relaxation failed.

- Settings in ini-file

  mlip:load-from - the file with an MTP. This should be a previously trained potential.

  select:load-state - the filename of the `mvs` file, containing the MaxVol state of the MTP (see –mvs-filename option in "select-add" and "train" commands). This is required by active learning.

  select:threshold - the γ_{tsh} on Figure 9.

  select:threshold-break - the Γ_{tsh} on Figure 9.

  select:save-selected - the file containing configurations from the `pre-selected` set (see Figure 9).

5.2.5 The "convert-cfg" Command

- Description

  Converting the configurations from one format no another.

- Supported formats

  `txt` (default): mlip textual format (`cfg` format)

  `bin`: mlip binary format (more faster I/O than with `cfg` format, but not human readable).

  `vasp-outcar`: only as input; VASP versions 5.3.5 and 5.4.1 were tested

  `vasp-poscar`: only as output; when writing multiple configurations,

POSCAR0, POSCAR1, etc. are created.
`lammps-dump`: only as input. Only lattice, atomic positions and types are saved. Only cartesian coordinates are processed.
`lammps-datafile`: only as output. Can be read by read_data from lammps. Multiple configurations are saved to several files.

- Input
  input-filename - the file containing configurations to be converted.

- Output
  output-filename - the file containing converted configurations.

- Options
  –input-format=*string*: format of the input file
  –output-format=*string*: format of the output file
  –append: opens output file in append mode
  –last: ignores all configurations in the input file except the last one (useful with relaxation)
  –fix-lattice: creates an equivalent configuration by moving the atoms into the supercell (if they are outside) and tries to make the lattice as much orthogonal as possible and lower triangular
  –save-nonconverged: writes configurations with nonconverged VASP calculations, otherwise they are ignored. Used only while reading from OUTCAR.
  –absolute-elements: writes absolute atomic numbers into the cfg file instead of 0,1,2,.... Disabled by default, used only while reading from OUTCAR.
  –elements-order=18,22,46,... atomic numbers separated with commas in the order as they are in POTCAR. Used only while converting to POSCAR.

5.2.6 The “self-test” Command

This command launches a self-testing utility with several dozens of tests. You can see the result of each test. If everything works as it should, all tests should pass. This command can be executed in parallel to test the parallelization.

5.2.7 The “help” Command

This command requires one argument: the name of the command, and prints info about this command.

5.2.8 The “list” Command

This command takes no arguments and prints the list of available commands.

5.3 Examples

In this section the demonstration of the most basic usage cases of the MLIP package will be provided. Among them are bash scripts with train, relax, select-add commands, and LAMMPS scripts using MTPs in various scenarious of simulations with active learning.

5.3.1 Linux Bash Scripts with Binary MLIP

The example bash scripts are located in `/doc/examples/MTP_with_AL` folder.

1. Training and relaxation

   This folder contains two subfolders: `success` and `failure`. They contain similar files except the `mtp` and `mvs` files, which results in successful relaxations in one case and breaking according to the extrapolation grade in the other case.

   In each of them the script `run.sh` launches training of MTP on the training set and performs relaxation of few dozens of structures with the trained potential. In both cases the active learning approach is implemented. Inside `run.sh` two variables are declared, standing for path to `mlp` binary and for the number of cores which are used to run the MLIP commands. Before running examples, an `mlp` binary should be compiled. After the compilation is created in `/bin` folder.

The script `train.sh` performs the `train` command with a few options:

```
mpirun -n $1 $2 train pot_clean.mtp train.cfg
--energy-weight=1 --force-weight=1e-3 --stress-weight=1e-4
--max-iter=100 --trained-pot-name=Trained.mtp
--mvs-filename=state.mvs
```

Here the `mpirun -n $1 $2` stands for launching a `mlp` binary in parallel with such options as path to the file and the number of cores, e.g., `mpirun -n 4 ../../../bin/mlp`. For a serial execution you can remove the `mpirun -n $1` part.

Words `train pot_clean.mtp train.cfg` are the name of the command and two arguments, meaning the file with MTP to train and the file with the training set (see Section 5.2).

Words
`--energy-weight=1 --force-weight=1e-3 --stress-weight=1e-4`
are the weights for the energies, forces, and stresses in the loss function (see Section 3.1).

Words
`--mvs-filename=state.mvs` specify the file with the MaxVol state to be created.

Once the potential is trained and MaxVol state is initialized the relaxation of the configurations is started.

The script `relax.sh` launches the `relax` command with a few options:

```
mpirun -n $1 $2 relax relax.ini
--cfg-filename=to-relax.cfg
```

```
--save-unrelaxed=out/unrelaxed.cfg
--save-relaxed=out/relaxed.cfg
```

The `relax.ini` is the name of file with the settings. It contains various settings, among which the line

```
mlip                    mtpr
```

is a required setting for working with multicomponent MTPs.

The `mlip:load-from` setting means the file with MTP to be used.

The thresholds γ_{trsh} and Γ_{tsh} from Figure 9 and Section 3.6.3 are contained in `select:threshold` and `select:threshold-break` fields. The name of the `mvs` filename is in the `select:load-state` field.

The option `--cfg-filename=to-relax.cfg` contains name of the file with the structures to be relaxed, while
`--save-unrelaxed=out/unrelaxed.cfg`
and
`--save-relaxed=out/relaxed.cfg` are the files with configurations, which failed to relax and relaxed succesfully.

In the `success` folder almost every relaxation finishes successfully, and in `failure` folder all reaxations are terminated. You can find the corresponding `cfg` and `log` files in the `out` subfolders in each folder.

2. Selection of configurations with active learning

   This folder contains an example of performing selection of new configurations to enter the training set, based on the MaxVol criterion (see Section 3.6). The script `run.sh` contains only one "select-add" command with corresponding arguments (see Section 5.2). As a result of executing this command, a files with the names like `train_selected.cfg_0` will be created. Such files contain outputs of all processes and should be concatenated in case of parallel execution. These files contain configurations, which should be added to the training set `train_init.cfg`, according to the MaxVol criterion.

5.3.2 Integration with LAMMPS

LAMMPS can use MTP as an interatomic potential. For installation details please refer to the MLIP manual. In this section a few examples of joint usage of LAMMPS with MTPs for MD with active learning framework are described. The folder with examples is located in `/doc/examples/LAMMPS_thresholds`. As written in MLIP manual, the part of the LAMMPS input script corresponding to MLIP is:

```
pair_style mlip mlip.ini
pair_coeff * *
```

where `mlip.ini` is the file with settings. The main settings are similar to the ones described in "relax" command (see Section 5.2). There are four folders in `/doc/examples/LAMMPS_thresholds`: `break`, `select`, and `no_select` contain different scenarios of MD with active learning while `out` folder contains output files and logs. Each example can be launched by running a `run.sh` script from the corresponding folder. LAMMPS binary file compiled with MLIP user package should be present in the `/bin/` folder to run these examples. Next we describe each example.

- **no_select**

  This example demonstrates the usage of MTP without selection, i.e., without considering extrapolation grades of configurations, occurring in MD. The corresponding line in `mlip.ini` file which disables the selection is

  ```
  select      FALSE
  ```

  MTP provides energies, forces and stresses for all configurations. After running a `run.sh` script the only file in the `out` folder will be the `record.cfgs`—the file with recorded configurations. The recording is done according to the following lines

  ```
  write-cfgs                ../out/record.cfgs
  write-cfgs:skip           99
  ```

in the `mlip.ini` file. The upper settings contains name of the file with recorded configurations while the lower one specifies how often the configurations from MD are recorded. You may notice two fields with features in each configuration:

```
Feature   EFS_by MultiMTP17
Feature   ind 201
```

The feature `EFS_by` tells that energies, forces and stresses were calculated with the MTP having 17 coefficients ξ_α (see (24) in Section 3). The feature `ind 201` tells that a particular configuration is recorded on the 201th time step.

- **select**

  This example demonstrates usage of MTP with active learning resulting in selection of extrapolative configurations without breaking the MD (similar to the non-significant extrapolation case in "training and relaxation" C++ example). After running the `run.sh` script the `out` folder will contain a file with not only recorded configurations, but also with the selected ones—`ts.cfgs_0`. Among the features of configurations the feature
  `MV_grade`
  contains the extrapolation grade of a current configuration. Only configurations with extrapolation grade higher than specified in the line
  `select:threshold    1.5`
  from the `mlip.ini` file are put into the selected set. Specifying a higher threshold might result in selection of no configurations.

- **breaking**

  This example demonstrates usage of MTP with active learning resulting in breaking of an MD simulation due to significant extrapolation, similar to significant extrapolation case in "training and selection" C++ example. After running the `run.sh` script the warning "Breaking threshold exceeded" will be printed, as extrapolation grade for configuration exceeds the one specified in the line

```
select:threshold-break    1000
```

As a result the `out` folder will contain a `ts.cfgs_0` file with only one configuration. The file `record.cfgs` will be empty as the very first configuration is extrapolative and no configurations are successfully calculated by MTP.

5.3.3 Python Implementation (mlippy)

The MLIP Python package is called `mlippy`. The user manual with installation information and list of available functions can be found in `/doc/mlippy/mlippy.pdf` file. This section describes the basic examples of using the `mlippy` package. Folder with examples can be found in `/doc/examples_python/` folder. In each folder the file `test.py` contains the Python code to be executed, while the rest of the files are required for the execution or are generated.

Each folder contains one example. At the start of each `test.py` file there are import statements as well as common header lines:

```
comm = mpi4py.MPI.COMM_WORLD
rank = comm.Get_rank()
mlippy.initialize(comm)
```

which works both for parallel, e.g.,

```
mpirun -n 4 python3 test.py
```

and serial execution:

```
python3 test.py
```

If you have any troubles with MPI or if you want to remove it from the script, then only following line should left:

```
mlippy.initialize()
```

Next we describe each example in details.

- **train**

  This scripts performs training of the MTP on a dataset. The line
  `mlip = mlippy.mtp()`
  declares an instance of the `mtp` class from `mlippy` namespace, which is the representation of a machine-learning potential can be loaded/saved from/to file and used inside Python.

  The line
  `mlip.load_potential('pot.mtp')`
  initializes the potential with the file. Declaration and initialization can be done within one line:
  `mlip = mlippy.mtp('pot.mtp')`

  The line
  `ts = mlippy.ase_loadcfgs('train.cfg')`
  reads a file `train.cfg` with configurations in a *train.cfg* format and returns a vector of `ase.Atoms` objects (the configurations format for `ase` library).

  The Python dictionary `opts = {"max-iter":"20",...` contains options to be passed to the command executing next. The names of the options are self-explanatory and mostly coincide with a set of options for binary `mlip` commands. By the way, 20 iterations is way to little and this number is put for the sake of speed. At least 200 iterations are recommended, while the default number is 1000. In any case, the output of the optimization procedure is available, and the optimal number of iterations can be derived from this output. When the loss function (which is minimized by the `mlippy.ase_train` function) starts to decreasing slowly, the training becomes less effective. Also, the `"conv_tol":"1e-8"` parameter can be eased for a less tight convergence criterion. If no options are specified they are initialized with the default values.

The line

```
mlippy.ase_train(mlip,ts,opts)
```

stands for the training procedure itself. It requires passing of the `mtp` object to be trained and a vector of `ase.Atoms` configurations as a training set. The other arguments are self-explanatory and optional.

The line
`mlippy.ase_errors(mlip,ts,on_screen=True)`
calculates the errors of the trained MTP on the provided training set. The optional argument `on_screen` defines whether to print this information on the screen or not. Using of the `mlippy.ase_errors(...)` without printing on the screen is provided further, in the “filter_stress” example.

The line
`mlip.save_potential('Trained_py.mtp')`
writes the trained `mlip` object to the file.

- **select**

  This script performs selection of configurations with active learning approach. The line
  `mlip.load_potential('Trained.mtp')` loads a trained MTP (the selection is useful for trained MTPs only) from the file.

  The lines

```
train_init = mlippy.ase_loadcfgs('train_init.cfg')
train_vasp = mlippy.ase_loadcfgs('train_vasp.cfg')
```

  load the initial training set and a set with the new configurations from the files as required by the “select-add” command (see Section 5.2).

The options are

```
opts = {"threshold":"2",
"mvs-filename":"state.mvs",
"selection-limit":"20"}
```

which carry exactly the same meaning as in “select-add” command for binary `mlip`.

The line
`diff_py = mlippy.ase_select(mlip, train_init, train_vasp, opts)`
returns new configurations from the `train_vasp.cfg` file to be added to the training set. After the execution of this line a file `new_state.mvs` is created. It contains information about the MaxVol state (see Section 3.6) and is used by active learning while relaxation (both via C++ and Python versions of MLIP) and while MD in LAMMPS. It contains information about the training set and is used to determine the extrapolation grades of configurations.

The line
`mlippy.ase_savecfgs('diff_py.cfg',diff_py)`
saves the vector of `ase.Atoms` to the file in `cfg` format.

- **relax**

  This script performs a structural relaxation (see Figure 11) of configurations with MTP. The relaxation is done with the following command:
  `relaxed_py = mlippy.ase_relax(mlip,to_relax)`

  The relaxation can be perfomed with the MTP and MaxVol state specified either in the lines (by default):

  ```
  mlip.load_potential('failing.mtp')
  opts["load-state"]="failing.mvs"
  ```

which will result in exceeding the extrapolation threshold for each of the relaxations, or in the lines:

```
mlip.load_potential('relaxing.mtp')
opts["load-state"]="relaxing.mvs"
```

which will result in successfull relaxation of almost every configuration.

Here `mlip` is an MTP used for the relaxation, and `to_relax` contains configurations to be relaxed. Note, that active learning is enabled by the option `"select":"TRUE"`. In case of `"select":"FALSE"` the relaxation will be done without considering MaxVol grades and without the need of specifying the `"load-state"` option. If the MTP is poorly trained, or significant extrapolation occurrs (regardless of its tracking) the relaxation can also fail to converge.

The function
`mlippy.ase_relax(...)`
returns the configurations, which are successfully relaxed. For setting a correspondence between initial configurations and the relaxed ones the automatic numbering of configurations is added. The number of a certain relaxed configuration in the initial vector can be accessed as `relaxed_py[i].features['ID']` field.

- **convert**

  This script performs conversions from VASP OUTCAR to ase.Atoms and from ase.Atoms to VASP POSCARs. The line
  `cfgs = mlp.ase_loadcfgs('train_abs.cfg')`
  reads the configurations in `cfg` format as `ase.Atoms` vector. The energy of the second configuration read is:
  `print (cfgs[1].energy)`
  The line
  `order = [13,29,47]`
  contains the list with the order of elements in the implied POTCAR

file, which is then used in the function
`mlp.convert_ase2vasp(cfgs,'myposcar',order)`
to make a correct ordering in the POSCAR files. The number of created POSCARs equals to the number of configurations in `cfgs` vector, and their names are `'myposcar0'`, `'myposcar1'`, etc.

The OUTCAR can be read with the function
`vasp_cfg = mlp.convert_vasp2ase('OUTCAR_')`
returning an `ase.Atoms` vector containing all found configurations. The line
`mlp.ase_savecfgs('from_vasp.cfg',vasp_cfg)` then saves them in a `cfg` format.

- **filter_stress**

  This scripts demonstrates the example of an utility which can be created with `mlippy`. It goes through a database and separates the configurations: it selects configurations on which the MTP has MAE lower then the provided threshold.

  In lines

  ```
  mlippy.initialize()
  fname = sys.argv[1]  #filename to load cfgs from
  stress_trsh = (float)(sys.argv[2])
  cfgs = mlippy.ase_loadcfgs(fname)
  max_str = 0
  for i in range(len(cfgs)):
  for j in range(6):
  if (abs(cfgs[i].stresses[j])>max_str):
  max_str = abs(cfgs[i].stresses[j])
  print ("max. stress component is " + str(max_str))
  ```

  the initialization of `mlippy` is performed, as well as reading of the arguments. Then loop goes through the database and selects the biggest stress component (among 6 of them in the Voigt notation) and prints it.

The lines

```
mlp = mlippy.mtp()
mlp.load_potential('pot.mtp')
```

load the MTP which will be used for calculating the errors. Alternatively, these lines can be replaced by one:

```
mlp = mlippy.mtp('pot.mtp')
```

The lines

```
accepted = []
rejected = []
for i in range(len(cfgs)):
report = mlippy.ase_errors(mlp,[cfgs[i]])
if ((float)(report['Stresses:
Average absolute difference']) < stress_trsh):
accepted.append(cfgs[i])
else:
rejected.append(cfgs[i])
print (report['Stresses: Average absolute difference'])
mlippy.ase_savecfgs(fname + "_accepted",accepted)
mlippy.ase_savecfgs(fname + "_rejected",rejected)
```

do the actual separation of configurations according to the stress MAE. The line
`report = mlippy.ase_errors(mlp,[cfgs[i]])` checks approximation errors of the `mlp` on the configuration `cfgs[i]`. The returning value is a dictionary of strings with the following structure:

```
{'Energy: Maximal absolute difference': '0.051103',
'Energy: Average absolute difference': '0.051103',
'Energy: RMS     absolute difference': '0.051103',
'Energy per atom: Maximal absolute difference': '0.003194',
'Energy per atom: Average absolute difference': '0.003194',
'Energy per atom: RMS absolute difference': '0.003194',
'Forces: Maximal absolute difference': '0.209541',
'Forces: Average absolute difference': '0.160072',
'Forces: RMS absolute difference': '0.173226',
'Forces: Max(ForceDiff) / Max(Force)': '0.390101',
'Forces: RMS(ForceDiff) / RMS(Force)': '0.477335',
'Stresses: Maximal absolute difference': '5.006112',
'Stresses: Average absolute difference': '5.006112',
'Stresses: RMS absolute difference': '5.006112',
'Stresses: Max(StressDiff) / Max(Stress)': '0.907327',
'Stresses: RMS(StressDiff) / RMS(Stress)': '0.907327'}
```

In the condition

```
(float)(report['Stresses:
Average absolute difference']) < stress_trsh)
```

the checking of the numerical value of a stress MAE is performed.

5.3.4 MLIP Calculator for ASE

`mlippy` can be used not only as standalone package possessing some functionality from the C++ version, but as a calculator for `ase.Atoms` configurations [1]. The corresponding example is contained in `/doc/examples_Python/MLIP_Calculator` folder and demonstrates the relaxation of a binary system using the `MLIP_Calculator`. After the import statements, the initialization of the `MLIP_Calculator` class is done using the `mtp` object:

```
mlp = mlippy.mtp('Trained.mtp')
calc = mlippy.MLIP_Calculator(mlp, opts)
```

The options

```
opts = {"select":"FALSE",
"load-state":"state.mvs",
"save-selected":"selected.cfg",
"threshold":"2",
"threshold-break":"10",
"write-cfgs":"record.cfgs",
"write-cfgs:skip":"3"
}
```

are similar to the settings in `mlip.ini` file used in LAMMPS (see Section 5.3.2). The option `"select":"FALSE"` means not considering extrapolation grades, while `"select":"TRUE"` permits termination of the calculation if severe extrapolation occurrs, writes extrapolative configurations to the file with `"save-selected"` name, and requires an `mvs` file specified by the `"load-state"` option.

This means that the `calc` object will calculate energies, forces and stresses based on the MTP from the `'Trained.mtp'` file. The lines

```
b = 5.01
a = ase.Atoms('AgPd',positions=[(0,0, 0), (b/2, b/2, b/2)],
cell=[(0, b, b), (b, 0, b), (b, b, 0)],pbc=True)
a = a.repeat(2)
a.rattle(stdev=0.5)
```

create the FCC lattice populated with Ag and Pd atoms and shuffle their positions. The line

```
a.set_calculator(calc)
mlippy.ase_savecfgs('initial_ase.cfg',[a])
```

attach our calculator to the atomic system and save the initial geometry of the system co compare it with the relaxed one. The lines

```
dyn = BFGS(a)
dyn.run(fmax=1e-1)
```

actually launch the relaxation procedure with the stopping criterion for forces' amplitudes set to 0.1. While relaxation you will see the typical output and then the line
`mlippy.ase_savecfgs('relaxed_ase.cfg',[a])` saves the relaxed configuration to the file. Note, that if you change the chemical symbols in AgPd formula (e.g., to ZrPd) you will get an exception:

`Atomic number 40 is not present in the MTP potential!`,

which means that the `mtp` object used in `MLIP_Calculator` is not trained for such species. As shown in the last line:
`print (mlp.species_avail())`
you can view the species present in an `mtp` object with `mlp.species_avail()` function. To extend the species which `mtp` can calculate, provide a training set with desired species and train the instance of the `mtp` class on it.

6 Perspectives of this work: big picture and impact

Nowadays the atomistic simulation approach is applicable to a very small fraction of the processes which are of scientific and/or industrial interest. This is due to the fact that models providing sufficiently accurate calculations of energies/forces in atomistic systems (QM models involving calculations of the electronic structure) are very time-consuming, and these expenses scale cubically with the number of atoms, making few hundreds of atoms an upper bound for the size of the system being simulated. Though sometimes more approximate linear-scaling implementations can be used, their computational cost is still too high for many applications. At the same time, for an adequate simulation of certain processes (e.g., nucleation evens, cracks propagation) the required number of particles can reach millions. The workarounds like using less accurate models (e.g., empirical potentials) or making predictions based on simulations of smaller time- and length scales sometimes work well and can provide useful results, but these are by far not universal solutions.

One of the problems for which atomistic simulation approach can potentially provide radically innovative solution is the search for new materials. Traditionally, it is performed in a blind trial-and-error manner. Designing a new multicomponent alloy for a spacecraft or aircraft needs involves looking over dozens of alloying elements, trying to find their optimal mixture among practically infinite number of possible constitutions. Further burdened by long and costly process of production and laboratory testing of each particular sample, the design of a new alloy can take years or even decades.

Few decades ago a similar situation was true for the engineering design scale (see Figure 24): the amount of possible constructions (of a plane, a car, or an engine) which should be tested for performance by far exceeded the capacity of testing facilities for physical samples. Emergence of the approaches and software enabling computer simulations of engineering constructions performance in different conditions opened a new page in production technology: nowadays with a help of computer-aided design (CAD) the most unusual and counter-intuitive forms are created, which provide the best performance among almost infinite number of possible solutions. Moreover, due to much greater speed and much lower cost of computer simulations compared to physical experiments, the timespan from an idea to its implementation in hardware has shortened at least by one order. By now, CAD is an integral part of any manufacturing design process, used in production of the smallest parts like microchips as well as planes or rockets. While *in vivo* testing is still the ultimate stage, computer simulations provide useful insights greatly reducing the amount of physical tests required.

Not a long time ago the introduction of DFT with accurate exchange-correlation functionals (GGA and more accurate ones) has led to a remark-

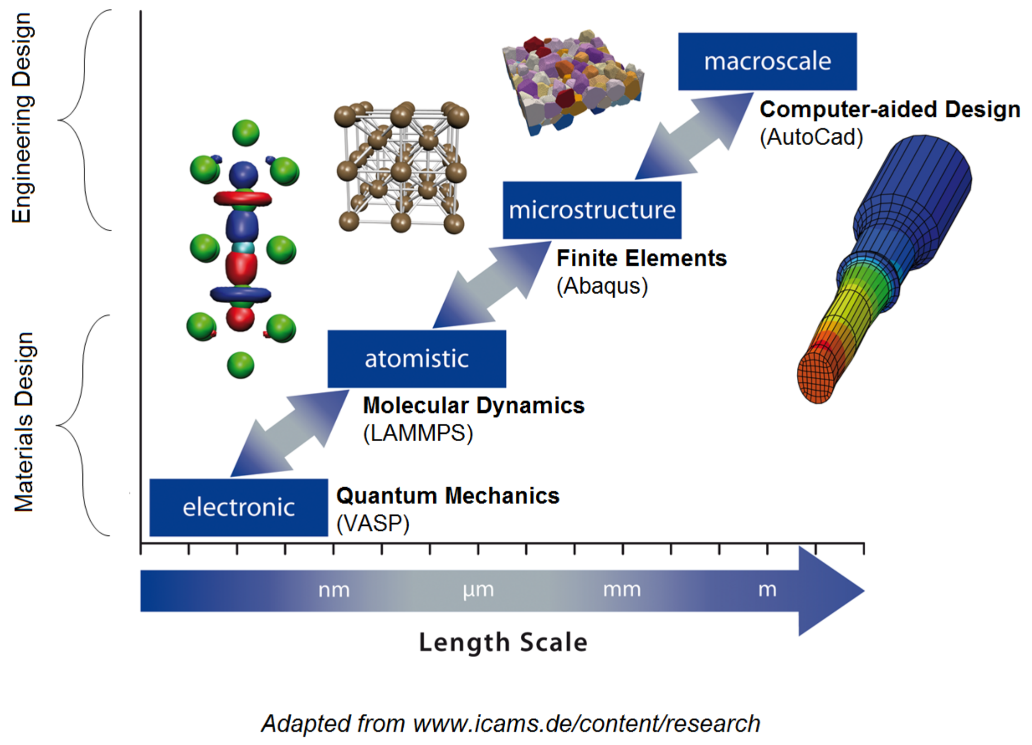


Figure 24: Theories and approaches (bold font) and examples of software (normal font) used for simulations at various length scales. At the scale of engineering design the presently available tools have already enabled fast and efficient simulations. At the materials design scale modern tools are often too computationally expensive to substitute empirical hands-on approach.

able breakthrough in computational materials science: though the speed of the DFT calculations is still by far prohibitive for many applications, DFT has proven itself to be very versatile and reliable tool, and it is widely used for studying broad classes of materials. On the other hand, the availability of machine-learning techniques (and, consequently, MLIPs) makes possible applying a surrogate modeling approach [25, 26] at the materials design scale (see Figure 24), allowing to simulate the behavior of atomistic systems without doing extensive number of DFT calculations. Instead, a small number of DFT calculations in carefully chosen points provides an input data for a MLIP, which is then used to study the behavior of the atomistic system. The accuracy of the resulting MLIP can only be slightly worse than that of the original DFT calculations, while the difference in computational speed usually constitutes three or more orders of magnitude (see Section 4.1.3). The calculations of high precision and high computational speed significantly extend the capabilities of the atomistic simulation approach, making it applicable to larger time and length scales. The parametrization of materials properties and the ability of machine-learning models to find the hidden relations between variables make it possible to construct a systematic approach for finding the materials with desired properties, as an alternative to the empirical trial-and-error approach. The simulations of both high accuracy and high speed, supported by the predictive power of machine-learning algorithms, give birth to a new paradigm called computer-aided materials design (CAMD). Within this paradigm the search for new materials is performed in a data-driven manner, analyzing the previous results and making predictions narrowing the area of the search. While the ultimate criterion is still an experiment, the predictions provided by computer simulation can significantly reduce the amount of experiments to be carried out.

The trend of CAMD is already recognized as an upcoming solution to the contemporary challenges of materials science, which is reflected in such strategic initiatives like Materials Genome Initiative in USA and European Materials Modeling Council in Europe. Such models of interatomic interaction as MLIPs are integral part of CAMD. Despite numerous examples of successful application of MLIPs for studying properties of different materials (see Section 4) their wide usage in industry is still a matter of a nearest future; effective, robust and universal enough software for materials design, which does not require a lot of expertise (and thus can be used not only by the scientists who developed it) is yet to be created.

As a natural development of the present research, the aim of creating methods and software for calculating various finite-temperature properties (phase stability, hardness, elastic moduli, etc.) of a broad range of materials can be pursued. Being implemented in a high-throughput and substantially automatized framework, the author believes it will make a great impact on the computational materials design field.

7 Conclusion

This thesis describes an approach to the construction and usage of a special type of interatomic interaction models—machine-learning interatomic potentials (MLIPs, see Section 2.4). The MLIPs I use in my research are of specific functional form based on local descriptors of atomic environment—the moment tensor descriptors (Section 3.4). These MLIPs are the extension of the originally proposed moment tensor potentials (MTPs, [74]) to the multiple component case. The MTPs proposed in this thesis incorporate non-linear functions of the fitting parameters and therefore I use a quasi-Newton Broyden – Fletcher – Goldfarb – Shanno local optimization algorithm in the training procedure (Section 3.5). The functional form of MTPs is systematically improvable, i.e., it can be extended with larger amount of fitting parameters at the cost of increased computational time. The unique functional form of MTPs allows for effective capturing of local interatomic interactions and therefore highly accurate reproduction of ab-initio results even with small amount of parameters and small training sets: see Section 4.2.

Another crucial component of the proposed methodology is the active learning (AL) approach (Section 3.6). It provides the numerical measure of extrapolation of the MTP for a certain configuration. This makes possible the so-called learning-on-the-fly: during an MD simulation the MTP is used to provide energies/forces/stresses only for configurations not far from the training domain, e.g., interpolative configurations. If an extrapolative configuration is detected, the MTP is retrained with additional relevant ab-initio data, extending the training domain and making the particular configuration interpolative, thus enabling reliable prediction of energies/forces/stresses. Additionally, active learning can be used to form the optimal training set out of fixed set of configuration: e.g., select 300 configurations out of 300 000, which carry the most relevant information for a machine-learning model. The selection is performed using unlabeled data, i.e., only positions and types of the atoms (Section 3.6.2, Section 4.2), thus allowing to compute ab-initio properties of the shortlisted configurations only after the selection procedure.

The combination of MTPs capable of accurate fitting and of the AL approach preventing extrapolation allows for effective screening of the atomistic structures in the crystal structure prediction problem (Section 4.1). Orders of magnitude faster structural relaxations done with MTPs provide opportunity to use much broader set of sample structures and therefore provide higher chances of detecting the stable structures: in Co-Nb-V (Section 4.1.2) and Al-Ni-Ti (Section 4.1.3) alloy systems MTP+AL approach allowed discovery of previously unreported ternary and binary structures. The overall speed-up in the stable alloys procedure reaches a factor of 100 (Section 4.1.3) while the formation energies and the geometries of the final structures are calculated with the DFT accuracy. The speed-up is due to effective and fast screening allowing to perform only the small fraction of calculations with the costly DFT method, rejecting all the others based on the MTP predictions.

Summing up, in this thesis I have proposed a way of constructing MLIPs for multicomponent alloys (and in addition successfully applied to the description of some organic molecules). The MLIPs (namely, MTPs) are capable of accurate fitting the energies/forces/stresses in atomistic systems, and together with the AL approach they form a powerful tool allowing for atomistic simulations with accuracy approaching the ab-initio one but orders of magnitude faster. The accuracy of the MTPs has been shown to be on the top level among other machine-learning models (Section 4.2), and there have been provided examples of succesfull application of the proposed methodology to solving problems previously done with DFT only (Section 4.1).

The limitations of the MTPs presently include atomistic systems where contribution of long-range interactions is essential, e.g., due to charge localization. Also, magnetic properties of the atoms are ignored in the current implementation.

All functionality is implemented in C++ and Python software packages, available via the following link:
https://gitlab.com/ashapeev/mlip-v2

References

[1] Atomic simulation environment: Calculators. https://wiki.fysik.dtu.dk/ase/ase/calculators/calculators.html.

[2] The novel materials discovery (NOMAD) laboratory, a European centre of excellence. http://nomad-repository.eu.

[3] A. Agrawal and A. Choudhary. Perspective: Materials informatics and big data: Realization of the “fourth paradigm” of science in materials science. *Apl Materials*, 4(5):053208, 2016.

[4] N. Artrith and A. M. Kolpak. Grand canonical molecular dynamics simulations of Cu–Au nanoalloys in thermal equilibrium using reactive ANN potentials. *Comput. Mater. Sci.*, 110:20–28, 2015.

[5] S. Bärthlein, G. L. W. Hart, A. Zunger, and S. Müller. Reinterpreting the cu–pd phase diagram based on new ground-state predictions. *Journal of Physics: Condensed Matter*, 19(3):032201, 2007.

[6] S. Bärthlein, E. Winning, G. L. W. Hart, and S. Müller. Stability and instability of long-period superstructures in binary cu–pd alloys: A first principles study. *Acta Materialia*, 57(5):1660–1665, 2009.

[7] 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.*, 104(13):136403, 2010.

[8] J. Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Phys. Chem. Chem. Phys.*, 13(40):17930–17955, 2011.

[9] J. Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *J. Phys. Condens. Matter.*, 26(18):183001, 2014.

[10] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98(14):146401, 2007.

[11] J. R. Boes, M. C. Groenenboom, J. A. Keith, and J. R. Kitchin. Neural network and ReaxFF comparison for Au properties. *Int. J. Quantum Chem.*, 116(13):979–987, 2016.

[12] V. Botu, R. Batra, J. Chapman, and R. Ramprasad. Machine learning force fields: Construction, validation, and outlook. *arXiv preprint arXiv:1610.02098*, 2016.

[13] V. Botu and R. Ramprasad. Learning scheme to predict atomic forces and accelerate materials simulations. *Phys. Rev. B*, 92(9):094306, 2015.

[14] N. J. Browning, R. Ramakrishnan, O. A. von Lilienfeld, and U. Roethlisberger. Genetic optimization of training sets for improved machine learning models of molecular properties. *The Journal of Physical Chemistry Letters*, 8(7):1351–1359, 2017.

[15] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *arXiv preprint arXiv:1802.09238*, 2018.

[16] S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, et al. Aflow: an automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012.

[17] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R. H. Taylor, L. J. Nelson, G. L. W. Hart, S. Sanvito, M. Buongiorno-Nardelli, et al. Aflowlib.org: A distributed materials properties repository from high-throughput ab initio calculations. *Computational Materials Science*, 58:227–235, 2012.

[18] M. S. Daw and M. I. Baskes. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Physical Review B*, 29(12):6443, 1984.

[19] S. De, A. P. Bartók, G. Csányi, and M. Ceriotti. Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics*, 18(20):13754–13769, 2016.

[20] J. Dennis and D. J. Woods. Optimization on microcomputers: The nelder-mead simplex algorithm. *New computing environments: microcomputers in large-scale computing*, 11:6–122, 1987.

[21] V. L. Deringer and G. Csányi. Machine learning based interatomic potential for amorphous carbon. *Phys. Rev. B*, 95:094203, Mar 2017.

[22] V. L. Deringer, C. J. Pickard, and G. Csányi. Data-driven learning of total and local energies in elemental boron. *Phys. Rev. Lett.*, 120(15):156001, 2018.

[23] P. E. Dolgirev, I. A. Kruglov, and A. R. Oganov. Machine learning scheme for fast extraction of chemically interpretable interatomic potentials. *AIP Advances*, 6(8):085318, 2016.

[24] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid dft error. *Journal of chemical theory and computation*, 13(11):5255–5264, 2017.

[25] A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide.* John Wiley & Sons, 2008.

[26] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. In *Proceedings of the royal society of london a: mathematical, physical and engineering sciences*, volume 463, pages 3251–3269. The Royal Society, 2007.

[27] M. Gastegger and P. Marquetand. High-dimensional neural network potentials for organic reactions and an improved training algorithm. *J. Chem. Theory Comput.*, 11(5):2187–2198, 2015.

[28] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: harmony search. *simulation*, 76(2):60–68, 2001.

[29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[30] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin. How to find a good submatrix. In *Matrix Methods: Theory, Algorithms, Applications*, pages 247–256. Word Scientific, 2010.

[31] A. Grisafi, D. M. Wilkins, G. Csányi, and M. Ceriotti. Symmetry-adapted machine learning for tensorial properties of atomistic systems. *Phys. Rev. Lett.*, 120(3):036002, 2018.

[32] K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev. Machine learning of molecular properties: Locality and active learning. *The Journal of Chemical Physics*, 148(24):241727, 2018.

[33] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. Von Lilienfeld, K.-R. Mu?ller, and A. Tkatchenko. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The journal of physical chemistry letters*, 6(12):2326–2331, 2015.

[34] G. L. W. Hart, S. Curtarolo, T. B. Massalski, and O. Levy. Comprehensive search for new phases and compounds in binary alloy systems based on platinum-group metals, using a computational first-principles approach. *Physical Review X*, 3(4):041035, 2013.

[35] G. L. W. Hart, L. J. Nelson, and R. W. Forcade. Generating derivative structures at a fixed concentration. *Computational Materials Science*, 59:101–107, 2012.

[36] J. D. Head and M. C. Zerner. A broyden—fletcher—goldfarb—shanno optimization procedure for molecular geometries. *Chemical physics letters*, 122(3):264–270, 1985.

[37] Y. Hinuma, T. Hatakeyama, Y. Kumagai, L. A. Burton, H. Sato, Y. Muraba, S. Iimura, H. Hiramatsu, I. Tanaka, H. Hosono, et al. Discovery of earth-abundant nitride semiconductors by computational screening and high-pressure synthesis. *Nature communications*, 7:11962, 2016.

[38] M. Hirn, S. Mallat, and N. Poilvert. Wavelet scattering regression of quantum chemical energies. *Multiscale Modeling & Simulation*, 15(2):827–863, 2017.

[39] B. Huang and O. A. Von Lilienfeld. Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity. *Journal of Chemical Physics*, 145(16), 2016.

[40] B. Huang and O. A. von Lilienfeld. The "dna" of chemistry: Scalable quantum machine learning with "amons". *arXiv preprint arXiv:1707.04146*, 2017.

[41] H. Huo and M. Rupp. Unified representation for machine learning of molecules and crystals. *arXiv preprint arXiv:1704.06439*, 2017.

[42] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. a. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[43] C. Jiang and B. P. Uberuaga. Efficient ab initio modeling of random multicomponent alloys. *Physical review letters*, 116(10):105501, 2016.

[44] S. R. Kalidindi and M. De Graef. Materials data science: current status and future outlook. *Annual Review of Materials Research*, 45:171–193, 2015.

[45] B. Kolb, L. C. Lentz, and A. M. Kolpak. Discovering charge density functionals and structure-property relationships with PROPhet: A general framework for coupling machine learning and first-principles methods. *Sci. Rep.*, 7(1), apr 2017.

[46] R. Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018.

[47] I. Kruglov, O. Sergeev, A. Yanilkin, and A. R. Oganov. Energy-free machine learning force field for aluminum. *Sci. Rep.*, 7(1):8512, 2017.

[48] D. D. Landis, J. S. Hummelshøj, S. Nestorov, J. Greeley, M. Dułak, T. Bligaard, J. K. Nørskov, and K. W. Jacobsen. The computational materials repository. *Computing in Science & Engineering*, 14(6):51–57, 2012.

[49] Z. Li, J. R. Kermode, and A. De Vita. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Phys. Rev. Lett.*, 114:096405, Mar 2015.

[50] N. Lubbers, J. S. Smith, and K. Barros. Hierarchical modeling of molecular energies using a deep neural network. *The Journal of Chemical Physics*, 148(24):241715, 2018.

[51] N. Lubbers, J. S. Smith, and K. Barros. Hierarchical modeling of molecular energies using a deep neural network. *J. Chem. Phys.*, 148(24):241715, jun 2018.

[52] A. O. Lyakhov, A. R. Oganov, H. T. Stokes, and Q. Zhu. New developments in evolutionary structure prediction algorithm uspex. *Computer Physics Communications*, 184(4):1172–1182, 2013.

[53] S. Manzhos, R. Dawes, and T. Carrington. Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces. *Int. J. Quantum Chem.*, 115(16):1012–1020, 2015.

[54] M. J. Mehl, D. Hicks, C. Toher, O. Levy, R. M. Hanson, G. L. W. Hart, and S. Curtarolo. The aflow library of crystallographic prototypes: part 1. *Computational Materials Science*, 136:S1–S828, 2017.

[55] B. Meredig, A. Agrawal, S. Kirklin, J. E. Saal, J. Doak, A. Thompson, K. Zhang, A. Choudhary, and C. Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[56] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, 2013.

[57] S. K. Natarajan, T. Morawietz, and J. Behler. Representing the potential-energy surface of protonated water clusters by high-dimensional neural network potentials. *Phys. Chem. Chem. Phys.*, 17(13):8356–8371, 2015.

[58] A. H. Nguyen, C. W. Rosenbrock, C. S. Reese, and G. L. W. Hart. Robustness of the cluster expansion: Assessing the roles of relaxation and numerical error. *Phys. Rev. B*, 96:014107, Jul 2017.

[59] C. Nyshadham, C. Oses, J. E. Hansen, I. Takeuchi, S. Curtarolo, and G. L. W. Hart. A computational high-throughput search for new ternary superalloys. *Acta Materialia*, 122:438–447, 2017.

[60] E. V. Podryabinkin and A. V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171–180, 2017.

[61] E. V. Podryabinkin, E. V. Tikhonov, A. V. Shapeev, and A. R. Oganov. Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. *arXiv preprint arXiv:1802.07605*, 2018.

[62] K. Rajan. Materials informatics: The materials "gene" and big data. *Annual Review of Materials Research*, 45:153–169, 2015.

[63] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1, 2014.

[64] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. Big data meets quantum chemistry approximations: the δ-machine learning approach. *Journal of chemical theory and computation*, 11(5):2087–2096, 2015.

[65] R. Ramakrishnan and O. A. von Lilienfeld. Machine learning, quantum mechanics, and chemical compound space. *arXiv preprint arXiv:1510.07512*, 2015.

[66] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[67] M. Rupp, R. Ramakrishnan, and O. A. von Lilienfeld. Machine learning for quantum mechanical properties of atoms in molecules. *The Journal of Physical Chemistry Letters*, 6(16):3309–3313, 2015.

[68] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[69] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (oqmd). *Jom*, 65(11):1501–1509, 2013.

[70] S. Sanvito, C. Oses, J. Xue, A. Tiwari, M. Zic, T. Archer, P. Tozman, M. Venkatesan, M. Coey, and S. Curtarolo. Accelerated discovery of new magnets in the heusler alloy family. *Science advances*, 3(4):e1602241, 2017.

[71] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 992–1002, 2017.

[72] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 992–1002, 2017.

[73] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017.

[74] A. Shapeev. Moment tensor potentials: a class of systematically improvable interatomic potentials. *Multiscale Model. Simul.*, 14(3):1153–1173, 2016.

[75] A. Shapeev. Accurate representation of formation energies of crystalline alloys with many components. *Computational Materials Science*, 139:26–30, 2017.

[76] A. V. Shapeev. Moment tensor potentials: a class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016.

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

[78] J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev, and A. E. Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24):241733, 2018.

[79] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[80] A. Sutton and J. Chen. Long-range finnis–sinclair potentials. *Philosophical Magazine Letters*, 61(3):139–146, 1990.

[81] W. J. Szlachta, A. P. Bartók, and G. Csányi. Accuracy and transferability of Gaussian approximation potential models for tungsten. *Phys. Rev. B*, 90(10):104108, 2014.

[82] A. Thompson, L. Swiler, C. Trott, S. Foiles, and G. Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *J. Comput. Phys.*, 285:316 – 330, 2015.

[83] M. C. Troparevsky, J. R. Morris, M. Daene, Y. Wang, A. R. Lupini, and G. M. Stocks. Beyond atomic sizes and hume-rothery rules: understanding and predicting high-entropy alloys. *Jom*, 67(10):2350–2363, 2015.

[84] S. Ubaru, A. Miedlar, Y. Saad, and J. R. Chelikowsky. Formation enthalpies for transition metal alloys using machine learning. *Physical Review B*, 95(21):214102, 2017.

[85] P. J. Van Laarhoven and E. H. Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[86] D. J. Wales and H. A. Scheraga. Global optimization of clusters, crystals, and biomolecules. *Science*, 285(5432):1368–1372, 1999.

[87] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2:16028, 2016.

[88] T. A. Wesolowski and J. Weber. Kohn-sham equations with constrained electron density: an iterative evaluation of the ground-state electron density of interacting molecules. *Chemical physics letters*, 248(1-2):71–76, 1996.

[89] Q. Wu, B. He, T. Song, J. Gao, and S. Shi. Cluster expansion method and its application in computational materials science. *Computational Materials Science*, 125:243–254, 2016.