Machine-learning approaches to model interatomic interactions in materials

M1 Internship Report
FunPhys Master

by

Carlos Rafael SALAZAR LETONA

Toulouse, June 2022

Supervisor: Dr. Julien LAM
Centre d'Élaboration de Matériaux et d'Etudes Structurales

Contents

Abstract

Machine learning (ML) methods have become more relevant in materials science over the last decade. They are now well recognized as effective methods for approximating extremely complex functions. Machine learning interaction potentials (MLIP) offer computation speeds close to empirical potentials, while having an accuracy close to the method used in training. In this project, the effects of the neural network (NN) architecture on its accuracy of prediction were studied. The machine learning framework n2p2 was used for the creation of potentials for gold systems from a Density-functional theory (DFT) database. The potential was then used together with Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) to calculate the surface energies and mechanical properties. These calculations were compared with values from DFT calculations as a measure of the agreement with DFT methods. Potentials were then trained using slightly different setup parameters in order to study their effect on the accuracy of the potential. The transferability of the potential was also investigated by training potentials with incomplete databases. It was found that the machine learning potential could be in good agreement with the DFT values, provided the setup parameters are properly chosen. Moreover, the potential showed good transferability, which is an important quality for the application of this potential in simulations. The results revealed that machine learning is not a one-size-fits-all solution, and that it still relies substantially on the setup parameters to properly minimize the errors with respect to the database.

1 Introduction

The simulation of complex systems of atoms and molecules is important in the fields of materials physics, chemistry, biology, among others. By performing numerical simulations, we are able to understand more in depth the molecular mechanisms responsible for macroscopic effects. Simulations are also helpful for studying systems for which it is too difficult or too expensive to perform experiments. In cases where an experiment is possible, they can indicate expected results, which can be helpful for preparing the experiment and the measurements that have to be made.

Material modelling at the atomistic level is concerned with the calculation of the potential energy surface (PES). The PES is a multidimensional function that provides the potential energy of a system as a function of the atomic positions [1], which is key to accessing the properties of a system. In materials science there are two main approaches to approximating the PES. On the one hand, electron-based methods are used for calculations at the electronic scales, they rely on quantum mechanics and are the most accurate models. However, they are also computationally expensive. On the other hand, *force field* methods are much faster, allowing for the simulation of larger systems, even reaching biological applications. But, they rely on empirical models, thus leading to a decrease in accuracy when compared to electron-based methods. In this project, machine learning is studied as a multi-scale method to bridge the electron-based and the *force field* methods.

1.1 Electronic-Based Methods

These methods make use of electronic Hamiltonians to evaluate the PES. Density-functional theory (DFT) is one of the most common methods used for electronic structure calculations [2]. DFT is based on extracting the physical properties of a system from its electron density $n(\vec{r})$, which is the spatial distribution function of all electrons described by the N-electron wave function [3]. This can be done because of the Hohenberg-Kohn theorems [4] [5]:

1. The ground state electron density is an injective function of the external potential and number of electrons, both of which determine the ground state of the wave function. In other words, there is a one-to-one mapping of the electron density and the wave function.

2. The variational principle can also be applied to the ground state electron density.

In this project, DFT calculations are used as a database for a machine learning algorithm. However, there are many more electronic-based methods, such as *ab initio* methods. These methods require only the atomic structure and fundamental laws of physics [2]. They are quantum mechanical approaches that do not need calibration against measured chemical parameters [6]. This is usually achieved by using the Hartre-Fock method, which relies on using

a single Slater determinant as ansatz for the all-electron wave function, and then exploiting the variational principle [3]. This, however, means that they are limited to a small number of atoms and small time scales due to their computational complexity.

Another example of electronic-based methods are semi-empirical methods. These methods rely on approximations of the Schrödinger equation, like the Hückel method. Because of this, they are also quantum mechanical methods, which consider the electronic structure of molecules. The approximations used depend on reference to experimental (or *ab initio*) quantities to produce the expected results [6].

1.2 Atomic-Based Methods

These methods are computationally simpler than electronic structure methods, as they ignore quantum mechanics calculations. Because of this, simulations at larger sizes and time scales are possible. In most atomic-based methods the motion of atoms is governed by Newtonian mechanics. Forces acting on the atoms are calculated using a *force field*, with which the PES is approximated. The *force field* can be expressed as a many-body expansion:

$$V(r_1, r_2, ...r_n) = \sum_{i}^{N} V_{1B}(r_i) + \sum_{i<j}^{N} V_{2B}(r_{ij}) + \sum_{i<j<k}^{N} V_{3B}(r_{ij}, r_{jk}, r_{ik}) + ... \quad (1)$$

where V_{nB} is the n-body interaction, r_i is the atomic coordinate, and r_{ij} are the distances between atoms i and j. There exist many classical interaction models to describe the n-body interactions. The most basic interaction to define a *force field* is the 2-body interaction. The most common 2-body interaction is the Lennard-Jones potential, which is capable of modelling attractive and repulsive interactions. It is commonly defined as [7]:

$$V_{LJ}(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \quad (2)$$

where r_{ij} is again the distance between atoms i and j, ϵ is the depth of the potential well, and σ is the distance at which the potential energy is zero. The ϵ and σ parameters can be adjusted to approximate different atomic species. Another example of an n-body interaction is the embedded-atom method (EAM) [8], which is widely used for modelling metals. In EAM, the metal is modelled as positively charged ions 'embedded' in a local electron density. Hence, the energy of the system is derived from an embedding energy and the ion core repulsion [9]. EAM approximates the local electronic density from the superposition of the atomic densities of surrounding atoms. Therefore, the total energy is given by:

$$E_{\text{tot}} = \sum_{i} F_i(\rho_{h,i}) + \frac{1}{2} \sum_{i \neq j} \Phi(r_{ij}) \quad (3)$$

where $F_i(\rho_{h,i})$ is the embedding function, $\Phi(r_{ij})$ is the short-range pair potential, r_{ij} is the distance between atoms i and j, and $\rho_{h,i}$ is the total host electron density at atom i. In this project, EAM calculations are used for comparison with classical interaction models.

1.3 Machine Learning Interaction Potentials

These methods are relatively new, and make use of machine learning algorithms. The main differences between classical and machine learning interaction potential (MLIP) approaches are the accuracy and flexibility possible with MLIPs. The mathematical form of MLIPs is more complex than most classical potentials, usually with hundreds of parameters. This allows MLIPs to better approximate the PES. This originality of the MLIP approach allows for bridging the gap between electronic and classical *force field* models and for performing large scale simulations with the accuracy of quantum based models.

The effectiveness of this methods has been proved on a variety of systems [10], [11], [12]. There are different methods for constructing MLIPs available, such as neural network potentials [13], kernel based potentials [14], gaussian approximation potentials [15], among others. In this project, a type of neural network potential is used, denominated high dimensional neural network potential [16].

1.3.1 Descriptors

A neural network (NN) takes as input a vector of numbers. Therefore, the atomic structure of a system must be broken down before being used as input for the NN. The descriptors are the functions G_i^μ whose job is to calculate the structural input for the neural network. If the cartesian position of the atoms is simply given as input, then the output will not be invariant under translation and rotation of the system. However, these properties are expected from a MLIP for it to be useful for the simulation of arbitrary system configurations. Ideally, the descriptors would produce values in a one-to-one correspondence with the atomic structure [1]. To tackle this problem, *atom centered symmetry functions* were proposed by Behler [17]. In this project, two *atom centered symmetry functions* are used, defined as:

$$G_i^2 = \sum_{j \neq i} \mathrm{e}^{-\eta(r_{ij} - r_s)^2} f_c(r_{ij}) \tag{4}$$

$$G_i^9 = 2^{1-\zeta} \sum_{\substack{j,k \neq i \\ j<k}} (1 + \lambda \cos\theta_{ijk})^\zeta \, \mathrm{e}^{-\eta((r_{ij}-r_s)^2 + (r_{ik}-r_s)^2)} f_c(r_{ij}) f_c(r_{ik}) \tag{5}$$

where G_i^2 is a radial symmetry function, G_i^9 is an angular symmetry function, r_{ij} represents the radial distance between atoms i and j, η is the width of the gaussian functions, r_s is the shift in the position of the gaussian function, ζ is

the angular resolution, $\lambda = \pm 1$ defines the position of the extrema of the cosine function, θ_{ijk} is the angle between three atoms and f_c is the cutoff function. For this project a polynomial and a cosine cutoff function were investigated, which are defined as:

$$f(x) = ((15 - 6x)x - 10)x^3 + 1 \tag{6}$$

$$f(x) = \frac{1}{2}\left(cos\left(\pi x\right) + 1\right) \tag{7}$$

$$f_c(r) = \begin{cases} f(x), & \text{for } r < r_c \text{ where } x := \frac{r}{r_c} \\ 0 & \text{for } r \geq r_c \end{cases} \tag{8}$$

where r_c is the cutoff radius. All of these parameters need to be chosen by the user. For this project, a cutoff radius of $r_c = 6.0$ Å was chosen. A list of the other parameters can be found in table 1, and the value of r_s is studied in the results section.

1.3.2 High Dimensional Neural Network Potential

Artificial Neural Networks (ANN) take inspiration from biology, as they are designed like a network of neurons in a brain. ANNs are networks formed by a large amount of simple and highly interconnected processing units known as neurons [18]. Neurons are grouped in layers within the network, and each neuron is connected to all the neurons in the next layer. The connections between neurons are described by weights and biases. An ANN can acquire knowledge through a learning process, generally adjusting the weights and biases in order to minimize an error function. High dimensional neural network potentials (HDNNPs) are a type of neural network potential (NNP) proposed by Behler and Parrinello [19]. This type of potential makes use of an *atomic* neural network as building block for a higher dimensional neural network. A diagram of a basic *atomic* neural network can be seen in fig 1. This basic network consists in one input layer receiving the atomic structure as input, one hidden layer for computations (it can be more than one), and an output layer consisting of one neuron which value provides the atomic energy for atom i. A network like this one (with one hidden layer) can be written down as a mathematical expression:

$$E_i = f_a^2\left[w_{01}^2 + \sum_{j=1}^{N_2} w_{j1}^2 f_a^1\left(w_{0j}^1 + \sum_{\mu=1}^{N_1} w_{\mu j}^1 G_i^\mu\right)\right] \tag{9}$$

where w_{ij}^k is the weight of the connection between neuron j in layer k and neuron i in layer $k-1$, w_{0j}^k is a bias, N_1 is the number of neurons in the input layer, N_2 is the number of neurons in the hidden layer, and f_a^k is the activation function for layer k. The atomic force can also be computed by calculating the derivative of this expression.

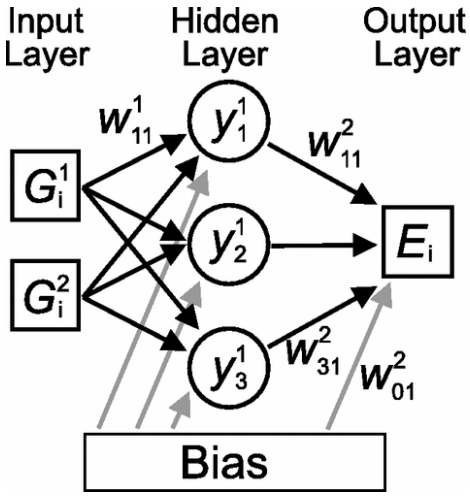


Figure 1: Diagram of an atomic neural network [19].

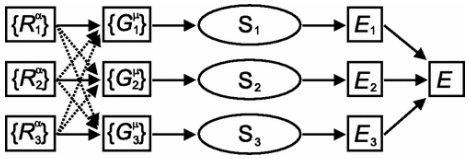


Figure 2: Diagram of a high dimensional neural network. R_i^α represents the cartesian coordinates of atom i, which are used to compute the symmetry function values G_i^μ used as input for the subnets S_i [19].

A high dimensional network is then constructed using as building blocks *atomic* networks for each atom in the system, represented as the subnet S_i. The high dimensional network can be seen in fig 2. HDNNPs are produced using this type of network. They are characterized by the use of many-body atom-centered symmetry functions and a set of *atomic* NNs which take the descriptor as an input, and produce the atomic energy as an output [16]. The goal of the high dimensional network is to represent the total energy of the system as a sum of atomic energy contributions as described by:

$$E = \sum_{i=1}^{N_{atoms}} E_i \tag{10}$$

In this project, the N2P2 software [20] was used to produce MLIPs for gold systems. The software follows the NN architecture presented previously to produce HDNNPs.

1.3.3 Training

The output produced by the NN previously described is computed using the weights of the connections. The training process consists in improving the output of the NN with respect to a reference value by adjusting the weights present in eq 9. The reference values are obtained from a database of calculations made using a different method. In this project, a database of DFT calculations for different gold systems was used. The DFT calculations provide values for the energy of the whole system, as well as the force vectors acting on each individual atom (3 force values per atom). This means that the database contains one energy value per system and $3N_a$ force values per system, where N_a is the number of atoms in the system. The database is then divided into train and test sets. The train set will be used as learning material for the algorithm, and the parameters will be adjusted to better reproduce the values in this set. The test set is not used in the training process, and is used to measure the capacity of the NN to interpolate systems not included in the training set. For all potentials used in this project, 10% of the database was selected for the test set, while the rest formed part of the train set.

The state of the neural network is described by a vector $\boldsymbol{w}$ containing the weight parameters. The training patterns are represented by the vector $\boldsymbol{x}_a$, where the index a enumerates the training patterns, and $\boldsymbol{y}^{\text{ref}}$ is the vector of reference output values corresponding to the input vector $\boldsymbol{x}_a$. The prediction of the neural network is represented by the vector $\boldsymbol{y}(t)$. The the error vector $\boldsymbol{\xi}(t)$ is then calculated as:

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, \quad \boldsymbol{y}^{\text{ref}} = \begin{bmatrix} y_1^{\text{ref}}(\boldsymbol{x}_a) \\ y_2^{\text{ref}}(\boldsymbol{x}_a) \\ \vdots \\ y_m^{\text{ref}}(\boldsymbol{x}_a) \end{bmatrix}, \tag{11}$$

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1(\boldsymbol{w}(t), \boldsymbol{x}_a) \\ y_2(\boldsymbol{w}(t), \boldsymbol{x}_a) \\ \vdots \\ y_m(\boldsymbol{w}(t), \boldsymbol{x}_a) \end{bmatrix}, \quad \boldsymbol{\xi}(t) = \boldsymbol{y}^{\text{ref}} - \boldsymbol{y}(t). \tag{12}$$

With the error vector, a cost function can be defined as $\Gamma = \sum_t \boldsymbol{\xi}^T(t)\boldsymbol{\xi}(t)$. The root mean square error (RMSE) is also defined as:

$$RMSE = \sqrt{\frac{1}{M}\sum_{a=1}^{M}(\boldsymbol{y}_a^{\text{ref}} - \boldsymbol{y}_a)^2} \tag{13}$$

where M is the number of training patterns. These two functions describe the disagreement of the NN predictions with respect to the reference values. During the training process both of these functions are minimized. Obtaining a good NN model requires the parametrization of the weights in eq 9. For that

purpose, numerous non-linear optimization methods can be employed including stochastic gradient descent (SGD) [21] and stochastic optimization [22], among others [23]. In this project, the extended Kalman filter (EKF) method was used, as it was demonstrated in ref [24] to have better convergence in this particular context.

The Kalman filter algorithm is an optimization technique that optimizes the estimate of the state of a linear dynamical system given observables from a history of data points [24]. It recursively calculates a better estimate of the weight parameters by using the error vector. Each estimate calculation is called an update. In the database there are both energy and force reference values, which can be used to calculate the error vector. Therefore, a weights update can be computed using energy or force values, in which case it could be an energy update or a force update. Each epoch of the training process consists of several updates.

It was shown by Blank and Brown [25] that the efficiency of the Kalman filter can be improved by selectively using information from the database. This idea relies on the assumption that patterns that are well reproduced by the NN can be skipped during an epoch. The relative error of a pattern with respect to the RMSE is used as a selection criteria.

1.4 Nanoparticles

Magnetic nanoparticles can have many biomedical applications, such as medical imaging, drug delivery, cancer therapy, etc [26]. However, common magnetic materials are not biologically compatible and oxidize with ease. An approach to tackle this problem is to cover magnetic nanoparticles with an inert and biologically compatible material. Gold is one of the materials being investigated for this purpose. Gold is biologically compatible, chemically inert and functions with several enzymes [27]. The synthesis of nanoparticles using gold and other magnetic materials is still improving and there are many questions to answer. Because of this, it is of great interest to perform simulations on these nanoparticles, for which machine learning methods could be of great help.

The goal of this internship project is to study the capacity of machine learning potentials to approximate gold PESs. The produced potentials will be compared with the classical EAM method, as well as with the DFT reference method. Furthermore, the effect of setup parameters on the calculation of physical properties will be studied. Finally, the transferability of the potential will be tested.

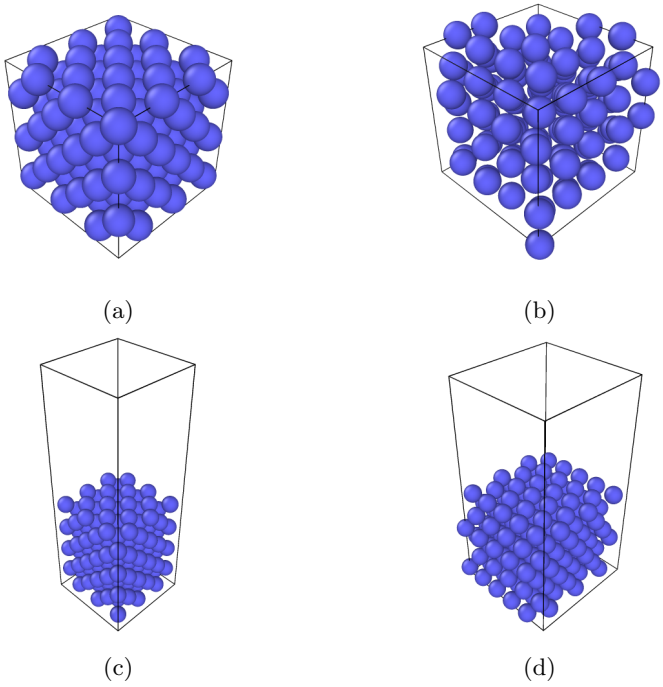


Figure 3: Figures (a) and (b) show the FCC and BCC bulk structures of gold, while figures (c) and (d) show the (100) and (111) surface structures of gold.

2 Methodology

2.1 Database

For this project, DFT calculations were used as a database for the training of the neural network. This database contained force and energy values for different gold systems. Included in the database were 30 (100) and 30 (111) surface structures as well as 25 bulk structures in BCC configuration, and 20 bulk structures in FCC configuration. The database and the values used for comparison come from ref [28].

2.2 Training

n2p2 [20] was the neural network framework used during this project. The layout of the network consisted of two hidden layers with 25 neurons each. The activation function used for the neurons in the hidden layers was a hyperbolic tangent, and a linear function for the output neurons. In order to check for the consistency of the NN potentials, 5 different potentials were created using different random seeds. The data points and error bars presented in this report

are obtained from the average and standard deviation of the calculations made using the different random seeds. 26 symmetry functions were defined using the parameters from ref [29], presented in table 1. The training was then performed for 100 epochs. Each epoch consisted of approximately 1000 updates (actual quantity depends on random numbers).

G^2	η	G^9	η	λ	ζ	G^9	η	λ	ζ
1	0.006428	1	0.000357	1	2	10	0.045357	-1	3
2	0.012856	2	0.005357	1	2	11	0.050357	-1	3
3	0.010357	3	0.010357	1	2	12	0.055357	-1	3
4	0.051424	4	0.015357	-1	2	13	0.60357	1	4
5	0.102848	5	0.020357	-1	2	14	0.65357	1	4
6	0.205696	6	0.025357	-1	2	15	0.70357	1	4
7	0.411392	7	0.030357	1	3	16	0.075357	-1	4
8	0.822784	8	0.035357	1	3	17	0.080357	-1	4
		9	0.040357	1	3	18	0.085357	-1	4

Table 1: Symmetry function parameters [29].

2.3 Physical Properties

Once the NNPs were obtained, they were used to compute physical properties of gold. The error of this computations with respect to the DFT calculations was used to monitor the accuracy of the system.

2.3.1 Mechanical properties

An energy and force minimization was performed on the FCC bulk in order to obtain the lattice spacing of the system. Once the system was minimized, the elastic properties were obtained by deforming the simulation box and measuring the change in the stress tensor. This, in turn, allowed to calculate the components of the elastic stiffness tensor. For a cubic system such as gold, the stiffness tensor can be described by 3 free parameters. With the components of the stiffness tensor, the bulk modulus could then be calculated using:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{bmatrix}, \quad B = \frac{C_{11} + 2C_{12}}{3} \tag{14}$$

In practice, the input script as given in LAMMPS was employed, and the methodology is further described in ref [30].

2.3.2 Surface energy

In order to calculate the surface energy, minimization was performed on surface slabs (100) and (111). From the minimization, the energy of the slab system could be obtained. Similarly for the bulk system, which was then divided by the number of atoms in the bulk system to obtain the energy per atom. The values were then used together with the area A of the surface to calculate the surface energy using the equation:

$$\gamma = \frac{1}{2A}\left(E^N_{\text{slab}} - NE_{\text{bulk}}\right). \tag{15}$$

where N is the number of atoms in the slab system, E^N_{slab} is the energy of the slab system, E_{bulk} is the energy per atom of the bulk system, and A is the area spanned by the slab system calculated using the dimensions of the simulation box.

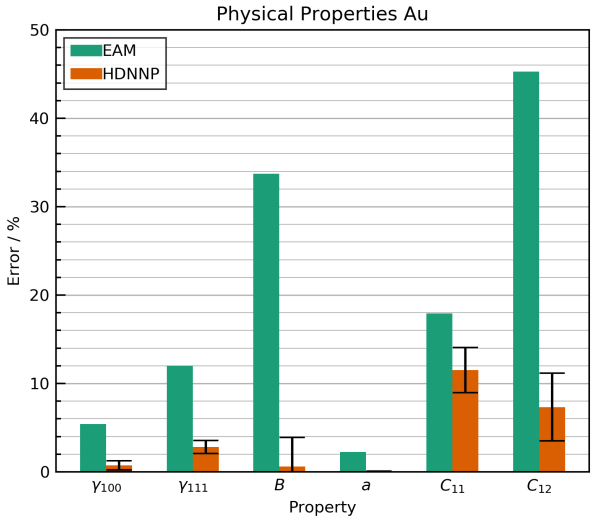


Figure 4: Error of different physical properties with respect to reference DFT values.

3 Results and Discussion

3.1 Comparison with EAM potential

One of the main goals of MLIPs is to improve on the accuracy of classical potentials. For this reason, calculations of physical properties were performed using the HDNNP and EAM potentials. The results are shown in fig 4. Even though EAM is a method designed for modelling metals, it can be observed that the values calculated with HDNNP have a much smaller error for all properties. This is as expected, and it is partly due to the choice of setup parameters used for the HDNNP, which will be discussed in the next subsections.

3.2 Cutoff function

To study the effects of the cutoff function on the accuracy of the HDNNP, two cutoff functions were tested. These are the polynomial and cosine cutoff functions as defined in equations 6 and 7. The potentials with both cutoff functions were trained using only force updates (i.e. energy values were ignored), and a value of $r_s = 0.0$ Å was chosen. The error of the physical properties with respect to DFT values was plotted and can be seen in fig 5. Both cutoff functions work good in this case, and in the case of elastic properties it is not possible to choose a favorite one. However, for surface energies the polynomial cutoff function approximates them better than the cosine cutoff function. For this reason, the rest of the potentials in this report are generated using the

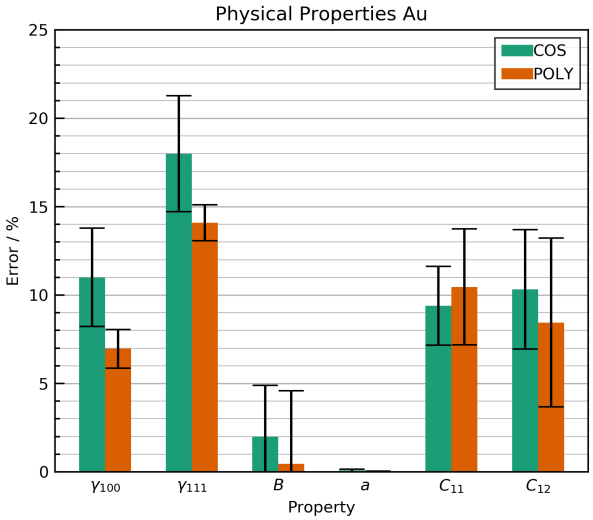


Figure 5: Error of different physical properties with respect to reference DFT values for the cosine (COS) and polynomial (POLY) cutoff functions.

polynomial cutoff function.

3.3 Dependence on r_s

Next, the effect of the symmetry function parameter r_s on test force RMSE and physical properties of the system was studied. In eq 4 the r_s parameter represents the shift of the gaussian functions that compose the symmetry function. The effect of using positive and negative r_s values on the symmetry function can be seen in fig 6. Different potentials were trained using r_s values in the range $[-r_c, r_c]$, where $r_c = 6.0$ Å is the cutoff radius. The potentials were trained using only force updates for each epoch. The results are shown in fig 7.

It can be seen that the error in physical properties becomes erratic in the region where $r_s > 2.0$ Å. The test force RMSE is shown in fig 7a. It can be observed that the error in surface energy seems to be correlated to the test force RMSE. Fig 6 shows the G^2 symmetry function for different r_s values. The test force RMSE seems to have a more stable behaviour when the symmetry function has a smaller slope, i.e. negative r_s values. The region between $r_s = -2.0$ Å and $r_s = 1.0$ Å was further investigated using 10 random seeds instead of 5 for greater precision, shown in fig 8.

In this range, no conclusion can be made about a dependence of physical properties on r_s. It is also observed that the test force RMSE remains more or less constant throughout the range $[-2, 0]$. For this reason, an arbitrary intermediary value of $r_s - 0.5$ Å was chosen to be used for the rest of the

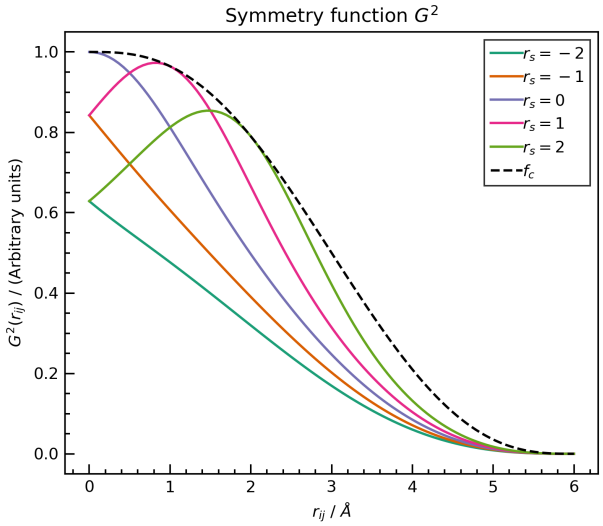


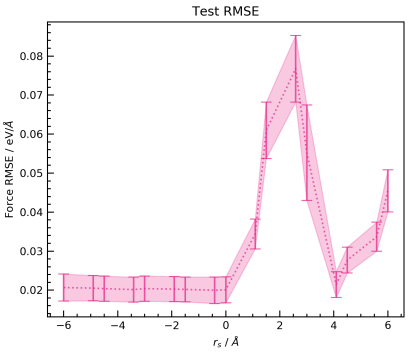
Figure 6: Cutoff function and symmetry functions for different r_s values.

potentials in this report.

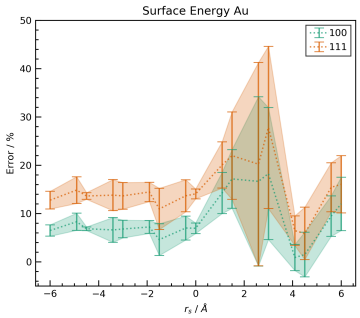
3.4 Training with energies

Next, the effect of the number of energy updates per training epoch was studied. The number of 1000 weight updates per epoch was kept in order to have similar training times. However, each of the 1000 weight updates can be based on an energy or a force value from the database, they can be either an energy or force update. n2p2 allows for granular control over the number of energy and force updates per epoch. Therefore, a force ratio can be defined as $ratio = updates_{force}/updates_{total}$, where the number of total updates is approximately 1000. Potentials were then trained using different force ratios ranging from 0 to 1. The rest of the updates will be energy updates, therefore the force and energy ratio add up to 1. The results can be seen in fig 9.

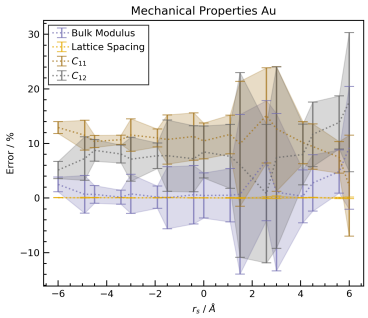
It can be seen that the error in surface energy significantly decreases for both (100) and (111) surfaces, getting very close to 0%. The machine learning potential clearly benefits from training with both force and energy updates. However, a training with only force updates still produces reasonable results, while a training with only energy updates has a large standard deviation. This can be due to the information made available by energy and force values. The database contains a total of 45240 force values, but only 105 energy values. For force ratios lower than 0.9 (energy ratios higher than 0.1) more than 100 energy updates will be used. As the number of energy values is limited, this means that force ratios lower than 0.9 will reuse energy values during each



(a)



(b)



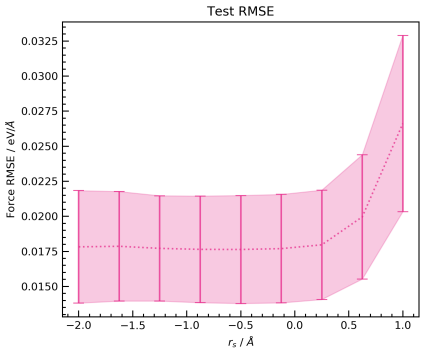
(c)

Figure 7: The test force RMSE is shown in plot (a). Plot (b) and (c) show the disagreement of the calculated physical properties with respect to the DFT values from ref [28]. It can be noticed that the plot of surface energies roughly resembles the plot of test force RMSE.

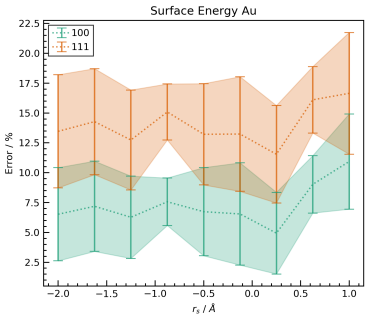
epoch. In principle, this is damaging to the training process, as it limits the novel information made available at each epoch.

Next, the effect of force ratio on mechanical properties was studied. The properties compared were the bulk modulus, lattice spacing and C_{11} and C_{12} components of the elastic stiffness tensor. The results are shown in fig 10. Out of all properties, the lattice spacing is the property best approximated by the machine learning potential. The bulk modulus benefits from a training using only forces, but the error is also considerably small when using forces and energies. For C_{11} and C_{12} no conclusion can be made. When the error for C_{11} or C_{12} decreases, the error for the other increases.

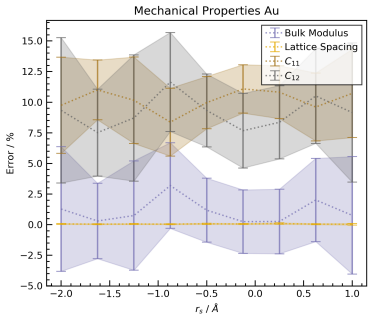
No conclusion can be made for an optimal force ratio value. The force ratio of 0.9 includes energy values without repeating them many times for the same epoch. This means that all energy values are included and there is more available



(a)



(b)



(c)

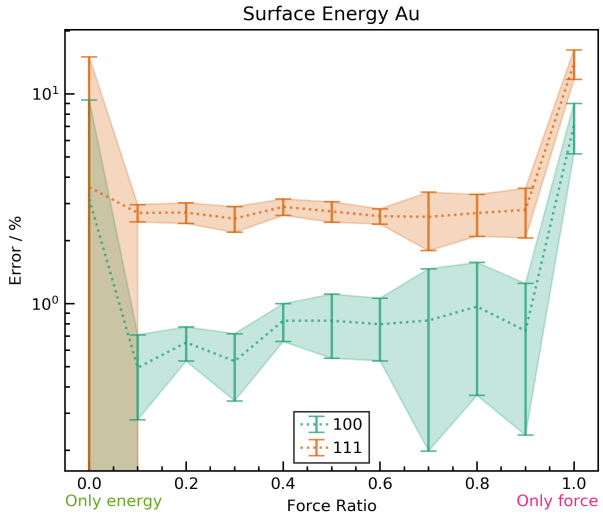
Figure 8: Plot (a) shows the test force RMSE as a function of r_s in the same range. Error of the calculated physical properties as a function of r_s in the range $[-2, 1]$ is shown in plot (b) and (c). No particular trend or optimal value can be identified within this range.

place for new force values to be included in each epoch. It is also recommended in ref [24] to use all energy values for training. Because of this, a ratio of 0.9 was used for the following potentials.

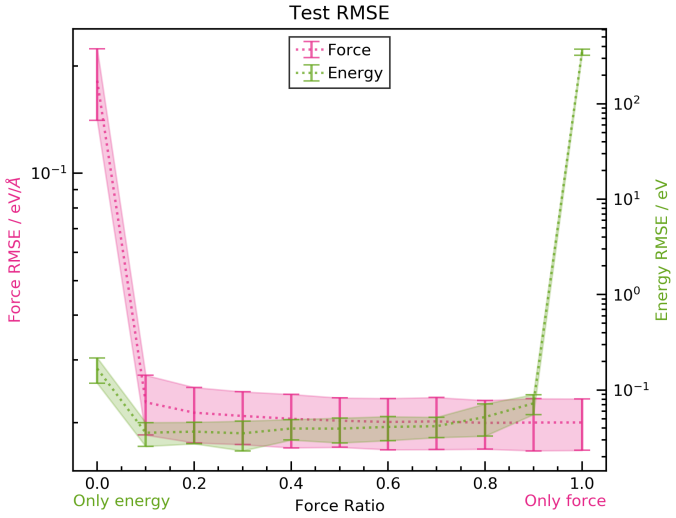
3.5 Transferability

In order to test for the capability of the machine learning potential to be used in more general applications, it is useful to study the effects when the database is limited. To analyze the transferability of the potential, the surface structures were removed from the database, and then the surface energies were calculated. The results for a complete database, a database without (100) surface structures, a database without (111) surface structures, and a database without any surface structures are shown in fig 11.

It can be seen that the training with the complete database has the smallest



(a)



(b)

Figure 9: In plot (a) the error of the surface energy is shown as a function of force training ratio in logarithmic scale. The ratio is defined as $ratio = updates_{force}/updates_{total}$, therefore a value of 0 means that all updates are energy updates, and a value of 1 means that all updates are force updates. The test energy and test force RMSE are shown in logarithmic scale in plot (b).

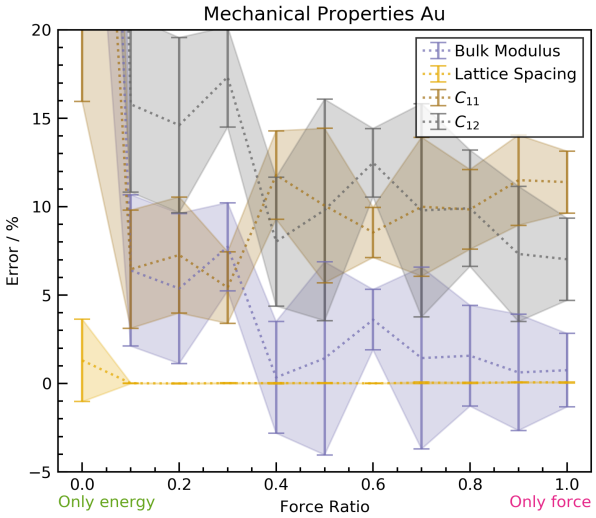


Figure 10: Error of different mechanical properties.

error percentage. The error in surface energy appears to be small when no surface structure is included in the database. However, the standard deviation is too high, and it could no longer be considered a useful approximation of a DFT calculation. When only one of the surface types is removed from the database, the error in energy of that surface type increases. However, the error in energy of the remaining surface type stays relatively similar. This means that as long as there is a surface type represented in the database, the other type can be successfully extrapolated.

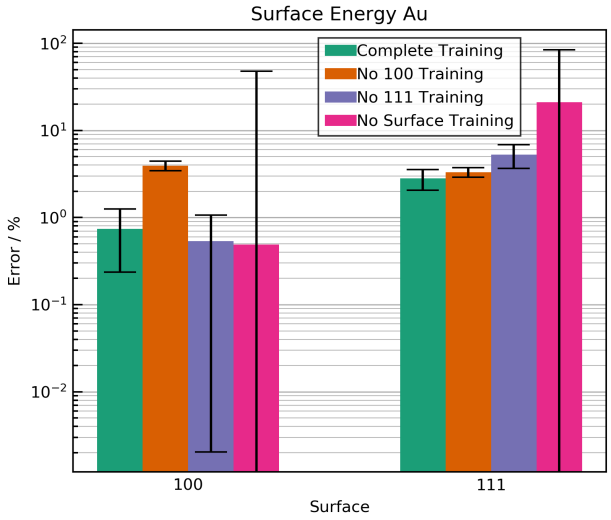


Figure 11: Error in surface energies for NN potentials trained on different databases. The *complete training* used all of the available database. For the *no 100 training* all (100) surface structures were removed from the database, equivalently for the *no 111 training*. Lastly, the *no surface training* used a database in which all surface structures were removed.

4 Conclusion and Outlook

Throughout this project, the capacity of machine learning potentials to approximate DFT calculations was assessed. In order to do this, physical properties were calculated with the machine learning potential and compared to the reference values from DFT calculations [28].

It is clearly proved that the HDNNP has a higher accuracy compared to the classical EAM potential. The error in physical properties was reduced when using the HDNNP, showing that HDNNP is a good multi-scale method to bridge the electronic and atomic scales.

Then, a study on the setup parameters of the HDNNP was performed. First, a comparison was made between two cutoff functions. It was found that both cutoff functions produce good results, but the surface energies are better approximated by the polynomial cutoff function.

The dependence of accuracy on r_s was studied next. It was found that negative r_s values provide reliable test force RMSE, even when studying more in detail the range $[-2, 1]$. It was then found that the surface energy plot slightly resembles the test force RMSE plot. The test force RMSE provides an indication on what the standard deviation of the surface energy will be. However, a good test force RMSE does not imply a small error in surface energies. This was

further illustrated in the range $[-2, 1]$. No clear trend or optimal value could be identified for the error in surface energies. These results show that there is a range of reasonable values for r_s in the range of $[-2, 1]$. And, it can be said that, in general, high r_s values produce higher test force RMSE and higher surface energy errors.

Then, the effect of energy and force training ratios on the potential was studied. As expected, it was found that the test force and test energy RMSE decrease as their respective ratio increased. However, when the force ratio is equal to 0.0 (only energy updates), the test energy RMSE was higher than for other ratios. This can be explained by the limitation of the energy data to describe the system. In total there were 105 energy values, one for each system in the database. This is minuscule compared to the 45240 force values. Energy values help the machine learning potential approximate the energy of the system, but they do not encode information on each atom. Therefore, force values are highly important. It was found that surface energy calculations clearly benefit from having access to both types of information. This is illustrated in fig 9a, where it can be seen that the error in surface energy decreases by a factor of 10 when the force ratio changes from 1.0 to 0.9. However, there is no clear decrease in the error as a function of force ratio for the rest of the plot. Instead, the force ratio has an effect on the standard deviation. For mechanical properties it is even more difficult to notice a trend. However, in the case of bulk modulus, there seems to be a better agreement for higher force ratios.

One of the most useful qualities of a machine learning potential is transferability. A database is computationally expensive to make, and therefore the number of systems included is limited. In this project, the transferability of the machine learning potential was tested by removing structures from the database, and observing the effect on surface energy. It was found that the machine learning potential is capable of correctly extrapolating a surface structure that is not included in the database. However, when all surfaces are removed the standard deviation increases too much for it to be considered a good approximation.

From these results, it is concluded that HDNNPs are capable of correctly approximating DFT calculations for a gold system. It was found that in spite of the flexibility of the algorithm the results still rely heavily on the chosen parameters. This may be problematic in some cases, and shows that machine learning is far from being a one-size-fits-all solution. In the future, this machine learning potential will be tested on more dynamic simulations to further validate against electronic-based methods.

5 References

[1] Jörg Behler. "Perspective: Machine learning potentials for atomistic simulations". In: *The Journal of Chemical Physics* 145.17 (2016), p. 170901. URL: https://doi.org/10.1063/1.4966192.

[2] Petr Klapetek. "Chapter 14 - Numerical Modeling Techniques". In: *Quantitative Data Processing in Scanning Probe Microscopy*. Ed. by Petr Klapetek. Micro and Nano Technologies. William Andrew Publishing, 2013, pp. 297–317. ISBN: 978-1-4557-3058-2. DOI: https://doi.org/10.1016/B978-1-45-573058-2.00014-0.

[3] Nongnuch Artrith. "High-dimensional neural network potentials for solids and surfaces". doctoralthesis. Ruhr-Universität Bochum, Universitätsbibliothek, 2013.

[4] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136 (3B Nov. 1964), B864–B871. URL: https://link.aps.org/doi/10.1103/PhysRev.136.B864.

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

[6] Errol Lewars. *Computational Chemistry: Introduction to the Theory and Applications of Molecular and Quantum Mechanics*. Boston, MA: Springer US, 2003. ISBN: 978-0-306-48391-2. URL: https://doi.org/10.1007/0-306-48391-2_1.

[7] J E Lennard-Jones. "Cohesion". In: *Proceedings of the Physical Society* 43.5 (Sept. 1931), pp. 461–482. URL: https://doi.org/10.1088/0959-5309/43/5/301.

[8] Murray S. Daw and M. I. Baskes. "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals". In: *Phys. Rev. B* 29 (12 June 1984), pp. 6443–6453. URL: https://link.aps.org/doi/10.1103/PhysRevB.29.6443.

[9] R.S. AVERBACK and T. DIAZ DE LA RUBIA. "Displacement Damage in Irradiated Metals and Semiconductors". In: *Solid State Physics*. Ed. by Henry Ehrenreich and Frans Spaepen. Vol. 51. Solid State Physics. Academic Press, 1998, pp. 281–402. DOI: https://doi.org/10.1016/S0081-1947(08)60193-9.

[10] Nongnuch Artrith and Jörg Behler. "High-dimensional neural network potentials for metal surfaces: A prototype study for copper". In: *Phys. Rev. B* 85 (4 Jan. 2012), p. 045439. URL: https://link.aps.org/doi/10.1103/PhysRevB.85.045439.

[11] Albert P. Bartók, Sandip De, Carl Poelking, Noam Bernstein, James R. Kermode, Gábor Csányi, and Michele Ceriotti. "Machine learning unifies the modeling of materials and molecules". In: *Science Advances* 3.12 (2017), e1701816. URL: https://www.science.org/doi/abs/10.1126/sciadv.1701816.

[12] Volker L. Deringer and Gábor Csányi. "Machine learning based interatomic potential for amorphous carbon". In: *Phys. Rev. B* 95 (9 Mar. 2017), p. 094203. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.094203.

[13] Matti Hellström and Jörg Behler. "Neural Network Potentials in Materials Modeling". In: *Handbook of Materials Modeling: Methods: Theory and Modeling*. Ed. by Wanda Andreoni and Sidney Yip. Cham: Springer International Publishing, 2020, pp. 661–680. ISBN: 978-3-319-44677-6. URL: https://doi.org/10.1007/978-3-319-44677-6_56.

[14] Christoph Scherer, René Scheid, Denis Andrienko, and Tristan Bereau. "Kernel-Based Machine Learning for Efficient Simulations of Molecular Liquids". In: *Journal of Chemical Theory and Computation* 16.5 (2020), pp. 3194–3204. URL: https://doi.org/10.1021/acs.jctc.9b01256.

[15] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. "Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons". In: *Phys. Rev. Lett.* 104 (13 Apr. 2010), p. 136403. URL: https://link.aps.org/doi/10.1103/PhysRevLett.104.136403.

[16] Jörg Behler. "First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems". In: *Angewandte Chemie International Edition* 56.42 (2017), pp. 12828–12840. DOI: https://doi.org/10.1002/anie.201703114.

[17] Jörg Behler. "Atom-centered symmetry functions for constructing high-dimensional neural network potentials". In: *The Journal of Chemical Physics* 134.7 (2011), p. 074106. URL: https://doi.org/10.1063/1.3553717.

[18] Melody Y. Kiang. "Neural Networks". In: *Encyclopedia of Information Systems*. Ed. by Hossein Bidgoli. New York: Elsevier, 2003, pp. 303–315. ISBN: 978-0-12-227240-0. DOI: https://doi.org/10.1016/B0-12-227240-4/00121-0.

[19] Jörg Behler and Michele Parrinello. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Phys. Rev. Lett.* 98 (14 Apr. 2007), p. 146401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.98.146401.

[20] Andreas Singraber, mpbircher, Sam Reeve, David W.H. Swenson, Jérémy Lauret, and philippedavid. *CompPhysVienna/n2p2: Version 2.1.4*. Version v2.1.4. May 2021. URL: https://doi.org/10.5281/zenodo.4750573.

[21] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. URL: https://arxiv.org/abs/1412.6980.

[23] W.H. Swann. "A survey of non-linear optimization techniques". In: *FEBS Letters* 2 (1969), S39–S55. ISSN: 0014-5793. DOI: https://doi.org/10.1016/0014-5793(69)80075-X.

[24] Andreas Singraber, Tobias Morawietz, Jörg Behler, and Christoph Dellago. "Parallel Multistream Training of High-Dimensional Neural Network Potentials". In: *Journal of Chemical Theory and Computation* 15.5 (2019), pp. 3075–3092. URL: https://doi.org/10.1021/acs.jctc.8b01092.

[25] Thomas B. Blank and Steven D. Brown. "Adaptive, global, extended Kalman filters for training feedforward neural networks". In: *Journal of Chemometrics* 8.6 (1994), pp. 391–407. DOI: https://doi.org/10.1002/cem.1180080605.

[26] Nhiem Tran and Thomas J. Webster. "Magnetic nanoparticles: biomedical applications and challenges". In: *J. Mater. Chem.* 20 (40 2010), pp. 8760–8767. URL: http://dx.doi.org/10.1039/C0JM00994F.

[27] Marie-Christine Daniel and Didier Astruc. "Gold Nanoparticles: Assembly, Supramolecular Chemistry, Quantum-Size-Related Properties, and Applications toward Biology, Catalysis, and Nanotechnology". In: *Chemical Reviews* 104.1 (2004), pp. 293–346. URL: https://doi.org/10.1021/cr030698+.

[28] Magali Benoit, Cyril Langlois, Nicolas Combe, Hao Tang, and Marie-José Casanove. "Structural and electronic properties of the Au(001)/Fe(001) interface from density functional theory calculations". In: *Phys. Rev. B* 86 (7 Aug. 2012), p. 075460. URL: https://link.aps.org/doi/10.1103/PhysRevB.86.075460.

[29] Troy D. Loeffler, Sukriti Manna, Tarak K. Patra, Henry Chan, Badri Narayanan, and Subramanian Sankaranarayanan. "Active Learning A Neural Network Model For Gold Clusters & Bulk From Sparse First Principles Training Data". In: *ChemCatChem* 12.19 (2020), pp. 4796–4806. DOI: https://doi.org/10.1002/cctc.202000774.

[30] *LAMMPS: Calculate elastic constants*. https://docs.lammps.org/Howto_elastic.html.