공학박사 학위논문

Metadynamics sampling for training machine learning potential

기계학습 퍼텐셜 개발을 위한 메타동역학 샘플링 기법

2020년 2월

서울대학교 대학원
재료공학부
유 동 선

Abstract

Machine learning potential is getting much attention as a promising computational tool that can give the accuracy close to that of quantum mechanical calculations with much lower cost that scales linearly with the number of atoms. The capability of machine learning potential was demonstrated through numerous applications to complex systems. However, a lack of fundamental understanding and the difficulties in constructing a training set are hindering a wide application of machine learning potentials. Despite the methodological advances in machine learning potentials, the conceptual foundation is still elusive due to the black-box nature of machine learning. The connection between machine learning potential and density functional theory, which forms the basis of machine learning potential, is not explicitly discussed. Besides, the construction of reliable machine learning potential requires a careful selection of a training set. The training set is usually selected by intuition and experience. Still, unexpected structures can emerge during the simulation, giving unreliable results or even catastrophic failures. Thus, a systematic method to construct a robust training set is desirable.

In this dissertation, we address two main issues hindering a wide application of machine learning potentials. First, starting from the nearsightedness principles, we formally derive transferable atomic energy within density functional theory, which is essential for all machine learning potentials that are based on the concept of locality. It becomes clear that the objective of machine learning potential is to learn the underlying atomic energy function from total energies. Using a classical potential as a reference, we show that machine learning potential is capable of learning the underlying atomic energy function when only total energies are informed. Through three simple examples, we demonstrate that machine learning potential is also prone to ad hoc energy mapping, where the potential gives an accurate prediction for the training set but learns markedly wrong atomic energy function. The implication for multi-component systems is also discussed. Next, we suggest a novel metadynamics method that can efficiently sample a wide range of local environments, enabling easier construction of a robust machine learning potential. We use a descriptor vector, which is an input to a machine learning model, as a collective variable for each atom. The total bias potential is a sum of atomic bias potentials. In this way, we can efficiently enhance sampling over local environment space for each atom rather than the configuration space of the whole system. We demonstrate three applications of the suggested metadynamics method. The metadynamics simulation can systematically sample a wide range of local environments that are energetically relevant. We apply the metadynamics sampling to develop general-purpose neural network potential for silicon and aluminum. Also, we show that the metadynamics simulation can be used to assess the stability of machine learning potentials and that the metadynamics sampling generates a robust training set, which improves the stability of machine learning potentials.

keywords: machine learning potential, neural network potential, metadynamics
student number: 2015-20848

Contents

Abstract **i**

Contents **iii**

List of Tables **vi**

List of Figures **vii**

1 Introduction **1**

1.1 Overview of machine learning potential 1

1.2 Goal of the dissertation 3

1.3 Organization of the dissertation 4

2 Theoretical background **5**

2.1 Density functional theory 5

2.1.1 Born-Oppenheimer approximation 5

2.1.2 Hohenberg-Kohn theorems 7

2.1.3 Kohn-Sham ansatz 8

2.1.4 Nearsightedness principles 10

2.2 Machine learning potential 11

2.2.1 Models 11

2.2.2 Descriptors 18

2.2.3 Input preprocessing 26

2.2.4 Training . . . 33

2.3 Metadynamics . . . 41

2.3.1 Adaptive Gaussians . . . 43

3 Atomic energy mapping 46

3.1 Introduction . . . 46

3.2 Transferable atomic energy within DFT . . . 47

3.3 Examples of atomic energy mapping . . . 52

3.3.1 Classical potential . . . 53

3.3.2 Si crystal example . . . 55

3.3.3 Si slab example . . . 58

3.3.4 Si nanocluster example . . . 63

3.4 Implication for multi-component system . . . 66

3.5 Summary . . . 69

4 Metadynamics sampling 70

4.1 Introduction . . . 70

4.2 Sampling methods . . . 73

4.3 Computational details . . . 81

4.4 Applications . . . 87

4.4.1 Catching unexpected failures . . . 89

4.4.2 Developing a general-purpose neural network potential . . . 92

4.4.3 Improving stability of potentials . . . 108

4.5 Summary . . . 112

5 Conclusion 113

Bibliography 115

Abstract (In Korean) 122

List of Tables

4.1 Metadynamics data set for general-purpose neural network potential for silicon . . . 82
4.2 Metadynamics data set for general-purpose neural network potential for aluminum . . . 83
4.3 Symmetry function parameters . . . 86
4.4 Comparison of various static properties between Stillinger-Weber potential and neural network potential for silicon . . . 96
4.5 Comparison of various static properties between density functional theory and neural network potential for silicon . . . 102

List of Figures

2.1 Structure of high-dimensional neural network potential 15

2.2 Shape of atom-centered symmetry functions 20

2.3 Schematic illustration of input preprocessing 29

2.4 Convergence of force RMSE against the training iteration with and without PCA and whitening 30

2.5 The trade-off between convergence speed and generalization error . . 31

2.6 Schematic illustration of metadynamics method 42

3.1 Correlation of atomic energy between EAM and NNP 54

3.2 Ad hoc energy mapping in silicon crystal example 56

3.3 Ad hoc energy mapping in silicon slab example 59

3.4 Connectivity analysis on silicon slab model 61

3.5 Convergence of atomic energy error during the training process 64

3.6 Atomic energy mapping in GeTe system 68

4.1 Evolution of potential energy of premade neural network potentials during metadynamics simulation 90

4.2 Sampling density of high-temperature molecular dynamics and metadynamics sampling (Stillinger-Weber potential) 95

4.3 Comparison of various static properties between Stillinger-Weber potential and neural network potential for silicon 97

4.4 Sampling density of high-temperature molecular dynamics and meta-dynamics sampling for silicon 100
4.5 Sampling density of high-temperature molecular dynamics and meta-dynamics sampling for aluminum 101
4.6 Comparison of various static properties between density functional theory and neural network potential for silicon 103
4.7 Energy correlation between density functional theory and neural network potential for silicon on various trajectories 106
4.8 Energy correlation between density functional theory and neural network potential for aluminum on various trajectories 107
4.9 Stability test with NNP-MD 110
4.10 Stability test with NNP-MetaD 111

Chapter 1

Introduction

1.1 Overview of machine learning potential

Machine learning potential (MLP) is getting much attention as a promising computational tool that can give the accuracy comparable to that of quantum mechanical calculations with much less computational cost that scales linearly with the system size. Various forms of MLP were suggested, including popular neural network potential (NNP)[1] and Gaussian approximation potential (GAP)[2]. The benefits of MLP were well demonstrated by numerous applications to complex systems. Owing to their promising future, the field of MLP started to grow explosively with the applications and methodological advances in the last few years.

Some notable applications include the crystallization behavior of $Ge_2Sb_2Te_5$[3], proton transfer at ZnO-water interface[4], a metastable structure search of Pt_{13} clusters[5], and an active site search of bimetallic catalyst for CO_2 reduction[6].

Methodological advances include feature selection using genetic algorithm[7, 8] and CUR decomposition[9], advanced machine learning architecture for complex systems such as stratified neural network[10], implanted neural network[11], mixture model[12], and weighting scheme to balance an inherent sampling bias[13]. Other forms of MLP such as spectral neighbor analysis potential (SNAP)[14], moment tensor potential (MTP)[15], deep tensor neural network (DTNN)[16], and gradient-domain machine learning (GDML)[17] were also suggested.

However, MLP is not widespread yet, considering the tremendous potential it has shown. This is partly because of hurdles to the adoption of MLP. Two major hurdles are a lack of fundamental understanding and the difficulties in constructing a training set.

Although the field of MLP is rapidly growing, the conceptual foundation of MLP is still elusive due to the black-box nature of machine learning. MLP represents the total energy as a sum of atomic energies, but one trains MLP on total energies of ab initio calculations (usually density functional theory). It is an unconventional machine learning problem. However, little attention has been paid to what MLP is actually learning and the impact of learning from the sum. Also, the connection of MLP to the density functional theory is not explicitly discussed yet, which is essential for the realization of MLP.

The second problem of constructing a training set arises from the *fragility* of MLP. Machine learning models are good at interpolation, but get completely lost when it comes to the extrapolation. MLP, which is based on a machine learning model without any physical base, suffers from the same problem. MLP gives results that are utter non-sense for the structures it was not trained for. This leads to the problem of simulation giving unreliable results (or even exhibiting catastrophic failures) as unexpected structures emerge. The training set should contain all configurations that can appear during the simulation in order to prevent such problems. The construction of reliable MLP, therefore, requires a careful selection of a training set (by intuition and experience) and even some trial and errors. This process is not only difficult but also time-consuming. Thus, a systematic and easy way to construct a robust training set is highly desirable.

If such hurdles are addressed and lowered, the development process of MLP could be accelerated, and MLP could be more widely adopted throughout the research community.

1.2 Goal of the dissertation

The main purpose of the dissertation is to address main difficulties in developing machine learning potentials mentioned in the previous section so that the machine learning potentials can be more widely adopted in the research. The dissertation also provides a gentle introduction to the machine learning potentials as we give the description of machine learning potential in-depth, including practical knowledge for the development of machine learning potentials.

We first deepen the understanding of machine learning potential by elucidating what machine learning potential is learning. We lay the foundation by defining atomic energy formally within density functional theory. It would help understanding riddling problems that arise during the development of machine learning potential and improving the quality of machine learning potential.

Next, we propose an improved sampling method that can be used to generate a training set with relative ease. The suggested method generates a robust training set such that the common problem of simulation failure could be averted. It would accelerate the development process significantly. The method can also be used to develop general-purpose potential, which is a tremendous work requiring much knowledge and intuition, by sampling a wide range of configurations automatically. It is also shown that the suggested sampling method can be used to measure the stability of premade potentials so that the quality of potentials can be assessed prior to the publication.

1.3 Organization of the dissertation

The dissertation is organized into five chapters. Chapter 1 is an introduction, which gives an overview of machine learning potentials as well as the goal of the dissertation. Chapter 2 introduces the basic theoretical backgrounds on the related subjects, such as density functional theory, machine learning potential, and metadynamics. The main results are divided into two chapters. Chapter 3 discusses the aspect of machine learning potentials related to their atomic energy mapping. First, the atomic energy is defined formally within density functional theory. Then, examples of atomic energy mapping are given. Additionally, the implication of atomic energy mapping for the multi-component system is discussed at the end of the chapter. Chapter 4 introduces sampling methods used for machine learning potentials and suggests a new metadynamics sampling method. The applications of the suggested metadynamics sampling are demonstrated and discussed. Finally, we summarize and conclude the dissertation in Chapter 5.

Chapter 2

Theoretical background

2.1 Density functional theory

2.1.1 Born-Oppenheimer approximation

Properties of matter are described from the interactions among electrons and nuclei, based on the theoretical foundation of quantum mechanics. The behavior of interacting electrons and nuclei can be described by Schrödinger equation. The many-body Hamiltonian for the system of electrons and nuclei is given by:

$$\begin{aligned}\hat{H} = &-\frac{\hbar^2}{2m_e}\sum_i \nabla_i^2 - \sum_I \frac{\hbar^2}{2M_I}\nabla_I^2 + \sum_{i,I}\frac{Z_I e^2}{|\mathbf{r}_i - \mathbf{R}_I|} \\ &+ \frac{1}{2}\sum_{i\neq j}\frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|} + \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J e^2}{|\mathbf{R}_I - \mathbf{R}_J|}\,,\end{aligned} \tag{2.1}$$

where $\mathbf{r}_j$ indicates the position of ith electron and $\mathbf{R}_I$ indicates the position of Ith nucleus (with charge Z_I and mass M_I). The first and the second terms are the kinetic energy of electrons and nuclei, respectively. The next three terms correspond to electron-nucleus, electron-electron, and nucleus-nucleus interactions, respectively. In general, the kinetic energy of nuclei can be regarded as small due to the large nuclear masses M_I. If the mass of nuclei is set to infinity, the kinetic energy of nuclei can be ignored, and the nuclei can be considered to have fixed positions, merely exerting external Coulomb potential on electrons. This is Born-Oppenheimer approximation.[18] Now the Hamiltonian in eqn 2.1 simplifies to:

$$\begin{aligned}\hat{H} &= \hat{T}_e + \hat{V}_{\text{ext}} + \hat{V}_{\text{int}} + E_{II} \\ &= -\frac{\hbar^2}{2m_e}\sum_i \nabla_i^2 + \sum_i V_{\text{ext}}(\mathbf{r}_i) + \frac{1}{2}\sum_{i\neq j}\frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|} + \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J e^2}{|\mathbf{R}_I - \mathbf{R}_J|}\,,\end{aligned} \tag{2.2}$$

where $\hat{T}_e$ is the kinetic energy of electrons, $\hat{V}_{\text{ext}}$ is the external potential acting on electrons, and $\hat{V}_{\text{int}}$ is the electron-electron interaction. E_{II} is the classical interaction energy between nuclei, which contributes to the total energy but irrelevant to the behavior of electrons (a constant term when the positions of nuclei are fixed). Here, the effect of nuclei of fixed positions is included in the external potential. Therefore, one can focus on the Hamiltonian of electrons with the positions of nuclei as parameters, simplifying the problems of interacting electrons and nuclei to the problem of interacting electrons in a static potential. Still, solving the equations for many-body Hamiltonian is nearly impossible for any practical systems of interest.

2.1.2 Hohenberg-Kohn theorems

The formulation of density functional theory is based on Hohenberg-Kohn theorems[19], which applies to the system of electrons and fixed nuclei whose Hamiltonian is given as eqn 2.2. The Hohenberg-Kohn theorems are as follows:

- **Theorem I:** For any system of interacting particles in an external potential $V_{\text{ext}}(\mathbf{r})$, the potential $V_{\text{ext}}(\mathbf{r})$ is uniquely determined by the ground state particle density.

- **Theorem II:** A universal functional for the energy $E[n]$ in terms of the particle density $n(\mathbf{r})$ can be defined, valid for any external potential. For any particular $V_{\text{ext}}(\mathbf{r})$, the exact ground state energy of the system is the global minimum value of the functional, and the density that minimizes the functional is the exact ground state density $n_0(\mathbf{r})$.

Since the Hamiltonian in eqn 2.2 is fully determined by the ground state density $n_0(\mathbf{r})$, it follows from theorem I that many-body wavefunctions of all states and hence all properties of the system are completely determined by the ground state density $n_0(\mathbf{r})$ alone. Theorem II indicates that the functional $E[n]$ alone is sufficient to determine the ground state density $n_0(\mathbf{r})$. One can minimize the total energy of the system with respect to the density to find the exact ground state density. The proofs of the theorems are gracefully simple, but not given in the dissertation.

The Hohenberg-Kohn theorems provide the basis for density functional theory. However, they give nothing about the actual form of functionals, and one is still left with the problems of solving a many-body system of interacting electrons, which is a formidable task.

2.1.3 Kohn-Sham ansatz

Kohn-Sham approach[20] replaces the original many-body problem by an auxiliary problem of non-interacting particles, which can be solved more easily. The Kohn-Sham ansatz is based on the assumption that the exact ground state density can be represented by the ground state density of an auxiliary system of non-interacting particles under certain potential, where the auxiliary system is described with an auxiliary Hamiltonian (assuming symmetric spin):

$$\hat{H}_{\text{KS}} = -\frac{\hbar^2}{2m_e}\sum_i \nabla_i^2 + \sum_i V_{\text{eff}}(\mathbf{r_i})\,. \tag{2.3}$$

The electron density $n(\mathbf{r})$ is given as a sum of densities of individual orbitals:

$$n(\mathbf{r}) = \sum_i |\psi_i(\mathbf{r})|^2\,, \tag{2.4}$$

and the kinetic energy T_s of independent particles is given as:

$$T_s = -\frac{\hbar^2}{2m_e}\sum_i \langle\psi_i|\nabla^2|\psi_i\rangle\,. \tag{2.5}$$

With the definition of Hartree energy, the classical Coulomb interaction energy of electron density $n(\mathbf{r})$ with itself:

$$E_{\text{Hartree}}[n] = \frac{e^2}{2}\int d\mathbf{r}d\mathbf{r}'\frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,, \tag{2.6}$$

the Kohn-Sham functional is written as:

$$E_{\text{KS}} = T_s[n] + \int d\mathbf{r}V_{\text{ext}}(\mathbf{r})n(\mathbf{r}) + E_{\text{Hartree}}[n] + E_{II} + E_{\text{xc}}[n]\,. \tag{2.7}$$

The terms involving $V_{\text{ext}}(\mathbf{r})$, $E_{\text{Hartree}}[n]$, and E_{II} are well defined, and the kinetic energy T_s is given as a functional of orbitals. This leads to equations for the system of non-interacting particles, which can be solved in a numerical manner, with all many-body terms incorporated into exchange-correlation energy E_{xc}. The exchange-correlation functional $E_{\text{xc}}[n]$ is merely the difference of kinetic and internal interaction energy of the true many-body system with the system of non-interacting particles, where the electron-electron interaction is replaced by Hartree energy.

Applying the variational principle leads to Schrödinger-like Kohn-Sham equation:

$$\left(-\frac{\hbar^2}{2m_e}\nabla^2 + V_{\text{eff}}(\mathbf{r})\right)\psi(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r})\,, \tag{2.8}$$

with

$$V_{\text{eff}}(\mathbf{r}) = V_{\text{ext}}(\mathbf{r}) + V_{\text{Hartree}}(\mathbf{r}) + V_{\text{xc}}(\mathbf{r}) \tag{2.9}$$

The Kohn-Sham equation is limited only by the approximation in the exchange-correlation functional. If the exact $E_{\text{xc}}[n]$ is known, the exact ground state density of interacting many-body systems can be found by solving Kohn-Sham equations for non-interacting particles. The exchange-correlation functional can be reasonably approximated by a semi-local functionals. The simplest yet powerful and popular form is linear density approximation (LDA)[20], where the exchange-correlation energy is approximated to that of homogeneous electron gas. In the linear density approximation, the exchange-correlation energy depends only on the electron density at the point. So, the exchange-correlation energy of LDA can be written as:

$$E_{\text{XC}}^{\text{LDA}} = \int \mathrm{d}\mathbf{r}\, n(\mathbf{r})\varepsilon_{\text{xc}}\left[n(\mathbf{r})\right]\,. \tag{2.10}$$

Another popular form is generalized gradient approximation (GGA)[21, 22], where not only local density but also the gradient of electron density are accounted. The exchange-correlation energy of GGA can be written as:

$$E_{\text{XC}}^{\text{GGA}} = \int \mathrm{d}\mathbf{r}\, n(\mathbf{r})\varepsilon_{\text{xc}}\left[n(\mathbf{r}), \nabla n(\mathbf{r})\right]\,. \tag{2.11}$$

2.1.4 Nearsightedness principles

The locality or *nearsightedness* of electronic matter is a well-accepted concept throughout the physics and chemistry despite of the non-locality of quantum mechanics.[23, 24] It is reasonable to assume that the property at one point can be considered to be independent to the change at distant points. The concept of chemical bonds and functional groups are also based on the locality.

The nearsightedness is a property of many-body systems. The density of an eigenstate at any point, strictly speaking, depends on the potential at all other points. However, in the many-body systems, the influence diminishes by the destructive interference between different independent particle eigenstates. It is known that a one-electron density matrix decays exponentially in an insulator or a metal at finite temperature (it follows a power law for metal at zero temperature). The one-electron density matrix is defined as:

$$\hat{\rho} = \sum_i |\psi_i\rangle\langle\psi_i| \,, \tag{2.12}$$

or

$$\rho\left(\mathbf{r}, \mathbf{r}'\right) = \sum_i \psi_i^*(\mathbf{r})\psi_i(\mathbf{r}') \,. \tag{2.13}$$

The density matrix is localized and vanishes at large separation when $T \neq 0$ or $T = 0$ in an insulator, even if the individual eigenfunctions ψ_i are extended over the entire system. Therefore, a static property at one point is only influenced by nearby nuclei and does not *see* the change of external potential at distant points (at fixed chemical potential), hence the name nearsightedness principle.

By utilizing the locality, order-N or $\mathcal{O}(N)$ methods are being developed,[25–27] where the computational cost scales linearly with the number of atoms. The details of order-N methods are not given here.

2.2 Machine learning potential

Ab initio calculations such as density functional theory are reliable and accurate. However, the simulation is limited in time and length scale as the computational cost is high and scales approximately to N_{atom}^3 (although $\mathcal{O}(N)$ methods are being developed, their application is still limited). For large scale simulations, classical potentials, or classical force fields, are favored due to low computational cost and linear scaling. However, they are limited in accuracy, and the development of potential for systems with complicated bonding nature is a daunting task.

Machine learning potential is a machine learning model that learns the relationship between structure and energy. There are many forms of machine learning potentials, varying from descriptors to models. They are appropriate to be used in dynamics simulation as they give continuous energy against the structure and are differentiable so that atomic forces can be obtained. If one trains machine learning potentials to the reference *ab initio* calculation results, one can obtain the accuracy near *ab initio* calculations with much lower computational cost that scales linearly to the number of atoms.

In this section, we give theoretical background for many aspects of machine learning potentials such as models, descriptors, input preprocessing, and training. The description is mostly focused on the neural network potential, which is used throughout the dissertation. However, most parts can be equally applied to other types of machine learning potential.

2.2.1 Models

High-dimensional neural network potential (HDNNP or NNP in short) suggested by Behler and Parrinello[1] and Gaussian approximation potential (GAP) suggested by Bartók and Csányi[2] are the first two models that gained popularity in the early age of machine learning potential. While other various models have been proposed,[14–

17, 28–30] HDNNP and GAP are still two most popular models used today.

The machine learning models can be categorized into either the neural-network-based model and the kernel-based model. HDNNP and GAP are categorized as a neural-network-based model and a kernel-based model, respectively.

Neural-network-based models utilize neural networks to predict atomic properties. The neural network is a function that has a large number of parameters (weights) and hence a very flexible shape. In the training process, one iteratively updates the weights of neural network such that the network gives accurate prediction of output properties compared to the reference. It requires a costly training process, and training parameters can affect the quality of potential. The evaluation cost mostly depends on the number of input descriptors and independent of the training set.

Alternatively, kernel-based models interpolate atomic properties as a linear combination of kernel function, which is a measure of similarity between the reference structure and the structure to predict. The coefficients are computed from linear algebra, and no iterative optimization process is required. The downside is that the cost of evaluation scales with the number of data set. (The evaluation cost can be reduced almost without the loss of accuracy by a delicate sparsification of the data set)

In this section, we describe in detail the high-dimensional neural network potential, which is used throughout the dissertation, and give a brief introduction to another popular model, Gaussian approximation potential.

High-dimensional neural network potential

A neural network (NN) is a mathematical model, which is inspired by the network of neurons in the brain. The neural network consists of nodes (neurons) and the directed weights (connections) between the nodes. A feed-forward neural network, or multi-layer perceptron, is a specific topology of the network where a number of neurons compose a layer, and the layers are densely connected in a certain direction, allowing the information to flow from one end (input layer) to the other end (output layer). The value of the node is computed as an activation function applied on the weighted sum of the previous layer:

$$x_i^{l+1} = f_a\left(b_i^l + \sum_{j=1}^{N^l} x_j^l \cdot w_{ij}^l\right) , \tag{2.14}$$

where x_i^l is the value of ith node in lth layer, N_l is the number of nodes in lth layer, w_{ij}^l is the weight connecting x_j^l and x_i^{l+1}, and b_i^l is a bias which adjusts the offset. f_a is the activation function, which gives non-linearity to the model. A sigmoid function $f_a(x) = 1/(1 + e^{-x})$, which is bounded in the range $[0, 1]$, is frequently used as the activation function. The activation function is not applied for the last layer (output layer) so that the output value is not bounded.

The power of the neural network lies in its flexibility. Although the functional form is fixed as eqn 2.14, the shape of the function varies depending on the weights. It is known that the neural network with a single hidden layer can approximate any continuous function with arbitrary accuracy given the number of nodes are sufficient.[31]

In the early applications of the neural network to fit the potential energy surface, a single feed-forward neural network is employed with the Cartesian coordinate, or interatomic distances were often used as input.[32–34] However, such approaches have serious limitations. First, the network cannot be applied to other systems with a different number of atoms. Second, the method is limited to only small systems with a few atoms. Lastly, the symmetry is not included in the model design, which can result in different energy for identical structures. For example, the permutation between two equivalent atoms could result in the change in energy. Several workarounds were suggested to solve the problems, but still, the application was limited.

In 2007, Behler and Parrinello suggested a high-dimensional neural network potential, which can be applied to the systems with an arbitrary number of atoms and includes appropriate symmetry invariance.[1] In HDNNP, the total energy is expressed as a sum of atomic energies:

$$E_{\text{tot}} = \sum_{i=1}^{N_{\text{atom}}} E_{\text{at}}\left(\mathbf{G}_i\right) \,, \tag{2.15}$$

where $\mathbf{G}_i$ represents the symmetry function vector of ith atom (see Section 2.2.2). Each atomic energy is evaluated from an atomic feed-forward neural network. The input to the *atomic* NN is a descriptor of the local chemical environment for each atom, which has a fixed dimension. The structure of high-dimensional neural network is illustrated in Fig. 2.1. The Cartesian coordinates of atoms are first transformed into symmetry function vectors, which describe the local environment. Atomic NN gives atomic energy for each atom as a function of symmetry function vector. Then, the total energy is a sum of atomic energies. Atoms with the same chemical species share the same atomic NN.

By utilizing the locality as eqn 2.15, HDNNP can be trained on small systems and then be applied to a much larger system with the same network. The symmetry invariance is incorporated into the descriptor, and any descriptor that satisfies the invariance condition can be used with the high-dimensional network structure. In most cases, atom-centered symmetry function, which was suggested with HDNNP, is used in conjunction with the high-dimensional network. The atom-centered symmetry function is described in Section 2.2.2.

The atomic forces are computed by differentiating the total energy with respect to the position of atoms:

$$F_{i,\alpha} = -\frac{\partial E_{\text{tot}}}{\partial R_{i,\alpha}} = -\sum_{j=1}^{N_{\text{atom}}} \sum_{s=1}^{N_G} \frac{\partial E_{\text{at},j}}{\partial G_{j,s}} \frac{\partial G_{j,s}}{\partial R_{i,\alpha}} \,, \tag{2.16}$$

where $R_{i,\alpha}$ is the coordinate of atom i ($\alpha = x, y, z$), $G_{j,s}$ is a sth component of symmetry function of atom j, and N_G is the dimension of a symmetry function vector.

The weight of the network needs to be optimized to give an accurate prediction. The training of high-dimensional neural network is similar to the ordinary feed-forward neural networks but more complicated for two reasons. First, HDNNP is

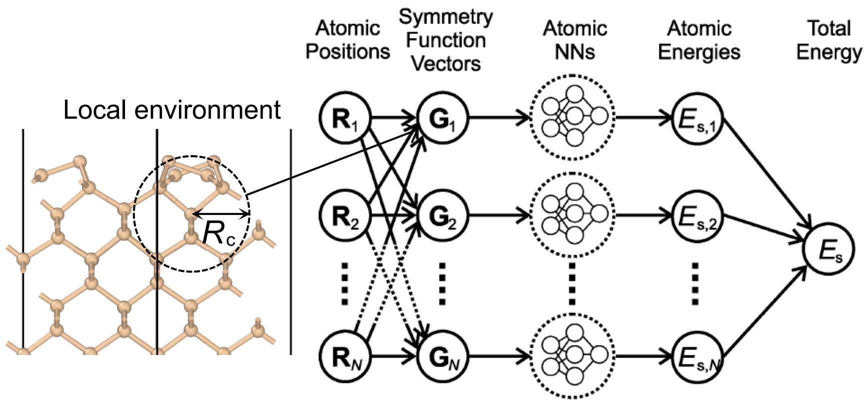


Fig. 2.1: Structure of high-dimensional neural network potential. For each atom, the local environment within cutoff radius R_{c} is encoded into a symmetry function vector $\mathbf{G}$ from atomic positions. Atomic NN takes the symmetry function as input and gives the atomic energy of the center atom as output. Finally, the total energy is represented as a sum of individual atomic energies.

trained over the sum of individual network outputs. Second, it is trained over not only energies but also over forces and stresses, which is related to the derivatives of the output property. The detailed discussion on the training process of the neural network potential is given in Section 2.2.4.

Gaussian approximation potential

Gaussian approximation potential (GAP) is suggested by Bartók and Csányi[2]. GAP is based on the Gaussian process regression (GPR). Suppose the atomic energy of an ith atom is written as a linear combination of basis functions:

$$E_i = \sum_h w_h \phi_h(\mathbf{d}_i)\,, \tag{2.17}$$

where $\mathbf{d}_i$ is a descriptor vector of the ith atom, and $\phi_h(\cdot)$ and w_h are an hth basis function and the corresponding coefficient, respectively. The prior distribution of the coefficients is given as $\mathcal{N}(0, \sigma_w^2)$. The covariance between atomic energies becomes:

$$\langle E_i E_j \rangle = \sigma_w^2 \sum_h \phi_h(\mathbf{d}_i)\phi_h(\mathbf{d}_j)\,, \tag{2.18}$$

where a kernel function can be defined, which is a measure of similarity between two configurations:

$$C(\mathbf{d}_i, \mathbf{d}_j) = \sum_h \phi_h(\mathbf{d}_i)\phi_h(\mathbf{d}_j)\,. \tag{2.19}$$

Now, the objective is to predict some property y (e.g., atomic energy or total energy) when observations $\mathbf{t}$ are given. The prior probability of observing $\mathbf{t}$ is given as:

$$P(\mathbf{t}) \propto \exp\left(-\frac{1}{2}\mathbf{t}^T\mathbf{C}^{-1}\mathbf{t}\right)\,, \tag{2.20}$$

where $\mathbf{C} = \langle \mathbf{t}\mathbf{t}^T \rangle$. Bayes' theorem states that the probability distribution of the prediction value y for new configuration under observations $\mathbf{t}$ is:

$$P(y|\mathbf{t}) = \frac{P(\mathbf{t}, y)}{P(\mathbf{t})}\,. \tag{2.21}$$

From eqn 2.20 and eqn 2.21, it can be shown that the mean of the probability distribution becomes $\bar{y} = \langle y\mathbf{t}\rangle^T \mathbf{C}^{-1}\mathbf{t}$.[35] This is the power of Gaussian process regression; one do not need to construct basis functions or obtain their coefficients in eqn 2.17. Only the kernel function and previous observations are required for the prediction. In addition, the uncertainty in prediction can be evaluated naturally since the prediction is based on the probability distribution. y can be anything where covariance can be defined, such as atomic energy, atomic forces, and total energy. For instance, the covariance between two total energies can be expressed as:

$$\langle E_I E_J \rangle = \left\langle \sum_{i\in I} E_i \sum_{j\in J} E_j \right\rangle = \sigma_w^2 \sum_{i\in I} \sum_{j\in J} C(\mathbf{d}_i, \mathbf{d}_j)\,, \tag{2.22}$$

where E_I is the total energy of an Ith structure.

Gaussian process regression is equivalent to a two-layer neural network with an infinite number of nodes and a hyperbolic tangent activation function for a specific kernel function. It is also equivalent to kernel ridge regression (KRR).

Smooth overlap of atomic positions (SOAP) is frequently used as a kernel function, which is described in Section 2.2.2.

2.2.2 Descriptors

The total energy of a system is invariant to several symmetry operations — translation and rotation of the whole system and permutation of the same chemical species. Therefore, it is desirable to have descriptors that are invariant to such symmetry operations. In this vein, several descriptors that satisfy the invariance condition are suggested to be used with machine learning potentials. Atom-centered symmetry function (ACSF) and smooth-overlap of atomic positions (SOAP) are popular choices for neural network potential and Gaussian approximation potential, respectively. There are also other types of descriptors, such as bispectrum descriptor and Zernike descriptor. In this section, we give a detailed description of ACSF and introduce SOAP briefly.

Atom-centered symmetry function

Atom-centered symmetry function (ACSF) was suggested by Behler et al.[36]. It is one of the most popular choices for neural-network-based models. It is specifically designed to meet the invariance condition, hence the name *symmetry* function. Among five functions suggested in the original paper, three functions are widely adopted: a radial symmetry function, G^{radial}, and two variants of angular symmetry function, $G^{\text{ang.n.}}$ and $G^{\text{ang.w.}}$. The radial symmetry function computes the value, based only on the distances between pairs, describing the radial distribution of neighboring atoms. On the other hand, the angular symmetry functions use not only distances but also angles between a triplet, giving the description of both radial and angular distributions.

The radial symmetry function of an atom i is defined as:

$$G_i^{\text{radial}} = \sum_j e^{-\eta(R_{ij}-R_{\text{s}})^2} f_{\text{c}}\left(R_{ij}; R_c\right) , \tag{2.23}$$

where R_{ij} is the distance between atom i and atom j, and f_c is a cutoff function. η, R_s, and R_c are parameters that determine the shape of the function. The function is invariant to global translation and rotation because only distances between atoms are used. It is also permutation invariant because it is calculated by summing the values over all pairs ij. η controls the decay rate, and R_s sets the center of Gaussian.

The cutoff function f_c is a smoothly decaying function, which is defined as:

$$f_\mathrm{c}(R_{ij}; R_c) = \begin{cases} 0.5\left[\cos\left(\frac{\pi R_{ij}}{R_\mathrm{c}}\right) + 1\right], & \text{if } R_{ij} < R_c \\ 0, & \text{otherwise} \end{cases} . \tag{2.24}$$

The main purpose of the cutoff function is to ensure the continuity of function value and its derivatives at the boundary of the cutoff sphere (when an atom enters or leaves the boundary). The cutoff function can be thought to additionally describe the decaying influence of an atom as the distance increases.

The angular symmetry functions are defined similarly to the radial symmetry function, but with the addition of a term related to the angle θ_{ijk} between triplet ijk (i on the center). The angular symmetry functions are defined as:

$$\begin{aligned} G_i^{\text{ang.n.}} =& 2^{1-\zeta} \sum_{j,k\neq i} (1 + \lambda \cos\theta_{ijk})^\zeta \, e^{-\eta\left(R_{ij}^2 + R_{jk}^2 + R_{ik}^2\right)} \\ & \times f_\mathrm{c}\left(R_{ij}; R_c\right) f_\mathrm{c}\left(R_{jk}; R_c\right) f_\mathrm{c}\left(R_{ik}; R_c\right) , \end{aligned} \tag{2.25}$$

$$\begin{aligned} G_i^{\text{ang.w.}} =& 2^{1-\zeta} \sum_{j,k\neq i} (1 + \lambda \cos\theta_{ijk})^\zeta \, e^{-\eta\left(R_{ij}^2 + R_{ik}^2\right)} \\ & \times f_\mathrm{c}\left(R_{ij}; R_c\right) f_\mathrm{c}\left(R_{ik}; R_c\right) . \end{aligned} \tag{2.26}$$

Again, angular symmetry functions are invariant to global translation and rotation because only distances and angles are used. They are also invariant to permutation as the angular symmetry functions for atom i are calculated by summing over all triplets ijk. $G^{\text{ang.n.}}$ is called *narrow* angular symmetry function, and $G^{\text{ang.w.}}$ is called *wide* angular symmetry function. The only difference is that two terms $e^{-\eta R_{jk}^2}$ and $f_c\left(R_{jk}; R_c\right)$ are multiplied to $G^{\text{ang.n.}}$ and absent on $G^{\text{ang.w.}}$. The additional terms diminish when θ_{ijk}, and thus R_{jk} increases. Therefore $G^{\text{ang.n.}}$ is limited in describing wide angles.

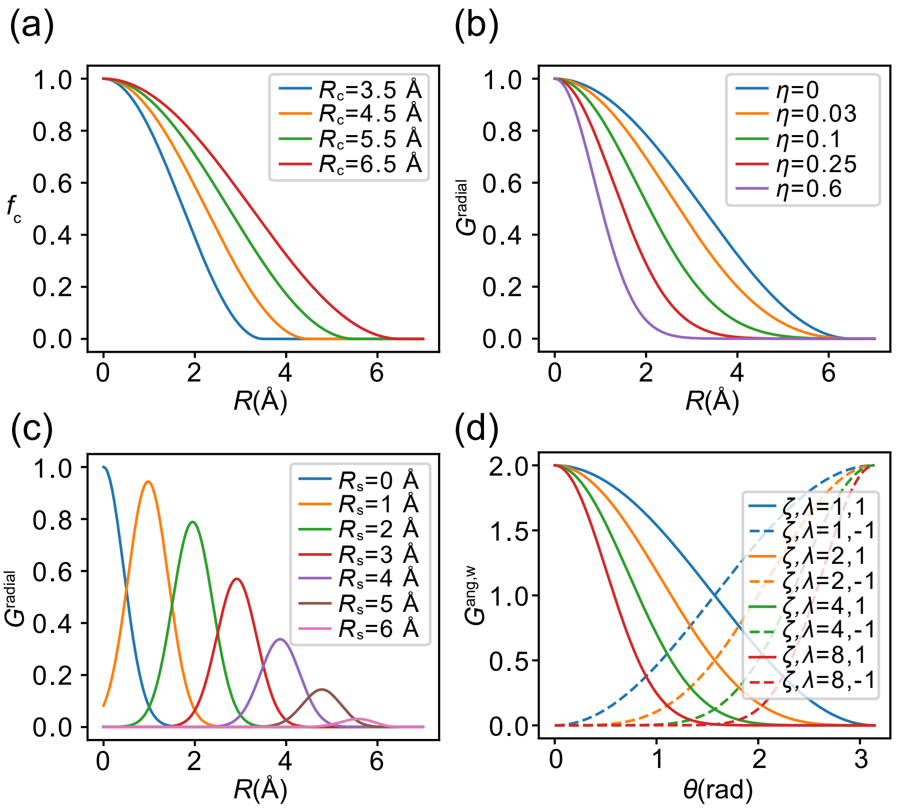


Fig. 2.2: Shape of atom-centered symmetry functions. (a) shows the shape of cutoff function f_c (eqn 2.24) with respect to the cutoff radius R_c. (b) and (c) show the shape of G^{radial} (eqn 2.23) against η and R_s, respectively. (d) shows the variation of $G^{\text{ang,w}}$ (eqn 2.26) with respect to ζ and λ. The shape of $G^{\text{ang,n}}$ is similar to $G^{\text{ang,w}}$, but it decays faster due to the decaying terms related to R_{jk}.

The parameter λ has the value of either 1 or -1 and controls the peak position along θ_{ijk} (0 or π). The parameter ζ gives angular resolution by controlling the sharpness of the peak. η similarly controls decaying rate against the distance, giving additional radial resolution to angular symmetry functions.

In order to describe the local environment of an atom, multiple symmetry function values are computed using different combinations of R_c, η, R_s, ζ, and λ values to obtain sufficient resolution. Then, a set of symmetry function values (a symmetry function vector) is used as an input to the machine learning models. The method to select an appropriate parameter set is discussed in the below.

For the system with multiple elements, a symmetry function is computed for a specific combination of elements. For example, in the A-B system, a radial symmetry function of A is calculated for either AA or AB pair. Similarly, an angular symmetry function of A is computed for one of AAA, AAB, and ABB triplet (A on the center). Although it is not necessary to have the same number of symmetry functions for each combination, typically, the number of radial symmetry functions scales with N_{element} and the number of angular symmetry functions scales with N^2_{element}. Therefore, the computational cost of execution (which depends mostly on the number of symmetry functions) approximately scales with N^2_{element}. Therefore, computational cost can be quite limiting for the system with a large number of elements. In order to deal with this quadratic scaling, weighted ACSF (wACSF) was suggested by Gastegger et al.[8]

In addition to the original forms suggested by Behler[36], some other variants, such as wACSF[8] and ANI-1 version[28], were proposed. But, we do not describe them in detail here.

Smooth-overlap of atomic positions

Smooth-overlap of atomic positions (SOAP)[37] is widely used in kernel-based models, including Gaussian approximation potential. For kernel-based methods, the descriptor itself is not essential. Only the kernel function, or similarity measure, constructed from the descriptors is required for the regression (see Section 2.2.1). SOAP takes an alternative approach where the kernel function is directly constructed, bypassing the construction of the descriptor. SOAP is designed such that it is invariant to symmetry operations, have well-defined limits, and changes smoothly with respect to Cartesian coordinates. First, the atomic neighbor density function is defined as a sum of Gaussians so that it varies smoothly to the atomic positions:

$$\rho(\mathbf{r}) = \sum_i \exp\left(-\alpha|\mathbf{r} - \mathbf{r}_i|^2\right) , \tag{2.27}$$

where i runs over the neighbors within the cutoff radius, and $\mathbf{r}_i$ is a vector from the central atom to its ith neighbor. If one defines a kernel function as the inner product of two atomic neighbor densities $[S(\rho, \rho') = \int \mathrm{d}\mathbf{r}\rho(\mathbf{r})\rho(\mathbf{r}')]$, it becomes invariant to translation or permutation. In order to make it invariant to rotations, $S(\rho, \rho')$ is integrated over all possible rotations $\hat{R}$:

$$k_n(\rho, \rho') = \int \mathrm{d}\hat{R} \left| \int \mathrm{d}\mathbf{r}\rho(\mathbf{r})\rho(\hat{R}\mathbf{r}') \right|^n , \tag{2.28}$$

where $n > 2$ since all angular information is lost when $n = 1$. Finally, SOAP kernel function is defined as:

$$C(\rho, \rho') = \left(\frac{k_n(\rho, \rho')}{\sqrt{k_n(\rho, \rho)k_n(\rho', \rho')}} \right)^\zeta , \tag{2.29}$$

where ζ is any positive integer, which controls the sensitivity of kernel to the change of atomic positions. Although SOAP was constructed from the similarity measure directly, it is closely related to power spectrum and bispectrum descriptor with Gaussian atomic neighbor density and a dot product covariance kernel.

Selection of a parameter set

For complete descriptors, one just increases the number of components to improve resolution systematically. However, for others (including ACSF), a parameter set must be selected somehow. It is important to choose an appropriate parameter set which can describe the local environment with sufficient resolution. If too few parameters are chosen, different local environments cannot be distinguished, resulting in high prediction error. However, using too many parameters increases the computational cost of execution and may hinder the training process. Therefore, it is important to find a balance between them.

For ACSF, several methods were suggested that can systematically improve the resolution by increasing the number of symmetry functions.[8, 38] In those methods, one first chooses the number of symmetry functions. Then, the parameters such as η and R_s are adjusted to provide a balanced coverage over the whole space. However, uniform coverage might not be the best representation. The methods describe above can provide the large pool of parameter set, from which one can choose the best subset. Several other methods were proposed to find the optimal parameter set.[7, 8, 38] We briefly discuss the parameter selection methods in the below.

CUR decomposition

CUR decomposition[39] is a feature selection method, which can select a subset of features that best represents the original matrix. Principal component analysis[40] can be used to reduce the number of features, but it gives the linear combination of original features. Therefore, one is still left with the problem of high computational cost as the number of features that one has to calculate remains the same. On the other hand, CUR-reduced features requires less computation as they are just a subset of original features.

CUR decomposition is a low-rank approximation using small number of columns and rows of the original matrix:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{CUR}\,, \tag{2.30}$$

where $\mathbf{C}$ and $\mathbf{R}$ are subsets of columns and rows of the original matrix $\mathbf{X}$.

For the purpose of parameter selection, $\mathbf{X}$ is the data set where each row vector is the symmetry function vector of an atom. Then, one finds the low-rank approximation, where only a small subset of columns is used. The selected columns are the parameters that best represent the full symmetry function vectors. (The same method can be applied on rows to select a subset of the training set)

One selects the columns one by one based on the importance score of each column. The importance score, or normalized statistical leverage score, is calculated as:

$$\pi_j = \frac{1}{k} \sum_{\xi=1}^{k} \left(\nu_j^{\xi} \right)^2 , \tag{2.31}$$

where ν_j^{ξ} is the jth coordinate of the ξth right singular vector and k is the number of features one selects. There can be different ways of selecting columns based on the importance score. We use the method described in Ref. [38]. First, one column with the highest importance score is selected. Then, all remaining columns are orthogonalized relative to the selected column. The procedure is repeated until the number of columns exceeds a certain number, or the error of low-rank approximation becomes lower than a threshold. The approximation error is defined as:

$$\epsilon = \frac{\|\mathbf{X} - \mathbf{CUR}\|_F}{\|\mathbf{X}\|_F} \tag{2.32}$$

The downside is that CUR decomposition does not take the regression model into account, so the actual performance may vary between models and not be optimal for the model. Still, it is a popular choice for selecting a subset of descriptors or the training set. We use CUR decomposition to select a parameter set from a large pool of parameters.

Genetic algorithm

Genetic algorithm, which is inspired by the evolution process, is widely used in various optimization problems. In the genetic algorithm, each individual has a genome, which is a parameter set, and genomes are subject to mutation and crossover operations. The parent genome for the next generation are chosen according to the fitness (performance) of each genome. Through the generation, the population evolves toward the higher fitness, so one can get parameter sets which perform well.

The genetic algorithm can be applied to find the best parameter set.[7, 8] The mutation and crossover operations are performed by exchanging parameters with the pool or by slightly changing the value of the parameter. The downside of the genetic algorithm is that it requires the evaluation of regression performance, thus requiring high computational cost. Therefore, the performance is usually evaluated with a smaller network or even linear models to approximate the performance in the computationally feasible way.

2.2.3 Input preprocessing

In order to train the machine learning model efficiently, it is recommended to preprocess inputs prior to the training. We scale symmetry functions and then transform the symmetry function vector with principal component analysis to improve the performance and convergence. In the following sections, we describe the preprocessing methods.

Scaling

The symmetry functions all have a different range of values, which depends on the training set. For instance, one component might have values in the range $[0, 10]$, while the other might have values in the range $[0, 10^{-3}]$. Even if two components both describe local environments in the same degree, machine learning model would have difficulty learning from the component with the smaller variance since the weight corresponding to the component has to be changed by greater amount during the training process. Therefore, we first scale each symmetry functions so that they all have a similar range.

There can be several ways to scale components. One can simply scale components so that the minimum becomes -1, and the maximum becomes 1. However, such scaling method could be too sensitive to the training set. It can also scale the component which virtually have zero value for all atoms, and amplify the numerical error. Another approach is to scale all components such that they all have zero mean and unit variance. It is less sensitive to the training set, but it is still inappropriate when all values are close to zero.

The other alternative is to scale with the reference value in the uniform gas.[38] One can calculate the value of the symmetry function assuming the atom is in the uniform background like jellium:

$$I^{\text{radial}} = 4\pi\rho_j \int dR_{ij} G^{\text{radial}}\left(R_{ij}\right) R_{ij}^2 \,, \tag{2.33}$$

$$I^{\text{ang.}} = 8\pi^2 \rho_j \rho_k \int dR_{ij} dR_{ik} d\theta_{ijk} G^{\text{ang.}}\left(R_{ij}, R_{ik}, \theta_{ijk}\right) R_{ij}^2 R_{ik}^2 \sin\left(\theta_{ijk}\right) \,, \tag{2.34}$$

where ρ_j is the average atomic density of the element of j (need not be precise). We then scale the symmetry function by the integral in eqn. 2.33 and eqn. 2.34 so that the value of each symmetry function is in the reasonable range. In this way, the scale of symmetry function does not depend on the training set while remaining in the reasonable range.

Principal component analysis

The symmetry function components suggested by Behler et al.[36] are highly correlated to each other. It is beneficial to decorrelate input features because it is known that the correlation between input features hinders the training process. Principal component analysis (PCA)[40] is a procedure that linearly transforms variables into uncorrelated variables called principal components.

Assume the data matrix $\mathbf{X}$ (dimension of $N_{\text{atom}} \times N_{\text{G}}$) is centered column-wise. The covariance matrix of $\mathbf{X}$ can be computed and diagonalized:

$$\Sigma^{xx} = \mathbf{X}^{\mathbf{T}}\mathbf{X} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^{\mathbf{T}} \,. \tag{2.35}$$

Therefore, one can define $\mathbf{Z}$ whose covariance matrix is diagonal. The principal component decomposition is given as:

$$\mathbf{Z} = \mathbf{X}\mathbf{W} \,, \tag{2.36}$$

$$\Sigma^{zz} = \mathbf{Z}^{\mathbf{T}}\mathbf{Z} = \mathbf{W}^{\mathbf{T}}\mathbf{W}\mathbf{\Lambda}\mathbf{W}^{\mathbf{T}}\mathbf{W} = \mathbf{\Lambda} \,. \tag{2.37}$$

The transformation can also be calculated using a singular value decomposition.

In addition, each principal component can be scaled to have the same variance, which is called *whitening*. Due to the strong correlation between symmetry function components, the variances of principal components vary several orders of magnitude. In order to learn from components with very small variance, some of the weights have to be very large. Therefore, the input data with such a wide range of variances would hinder the training process. The whitening process can be done by simply dividing each principal component by its standard deviation:

$$z_{(i)}^{\text{whiten}} = \frac{z_{(i)}}{\sqrt{\text{Var}\left[z_{(i)}\right]}}\,, \tag{2.38}$$

where z_i is ith principal component. The procedure of PCA is illustrated in Fig. 2.3 and the effect of PCA on the training is shown in Fig. 2.4. As can be seen in Fig. 2.4, applying PCA dramatically improves the convergence speed, and whitening further improves the speed while reducing RMSE at the same time. We note that it is better to add a small constant to the variance to suppress components with too small variances to be scaled up:

$$z_{(i)}^{\text{whiten}} = \frac{z_{(i)}}{\sqrt{\text{Var}\left[z_{(i)}\right] + \epsilon}}\,. \tag{2.39}$$

When we use eqn 2.38, we found that the neural network is much more vulnerable to overfitting. On the contrary, when we use eqn 2.39 with large ϵ, the convergence speed decreases (when ignoring the difference in scale, the $\epsilon \to \infty$ limit corresponds to no whitening at all). There is a trade-off between convergence speed and generalization error, which can be adjusted through ϵ (see Fig. 2.5). (eqn 2.38 with regularization techniques could prevent overfitting, but using eqn 2.39 resulted in the lower RMSE)

What we found is that the principal component with smaller variance shows a higher tendency to result in high generalization error. It can be explained qualitatively from a dynamic model of a simple shallow network. According to S. Advani et al.[41],

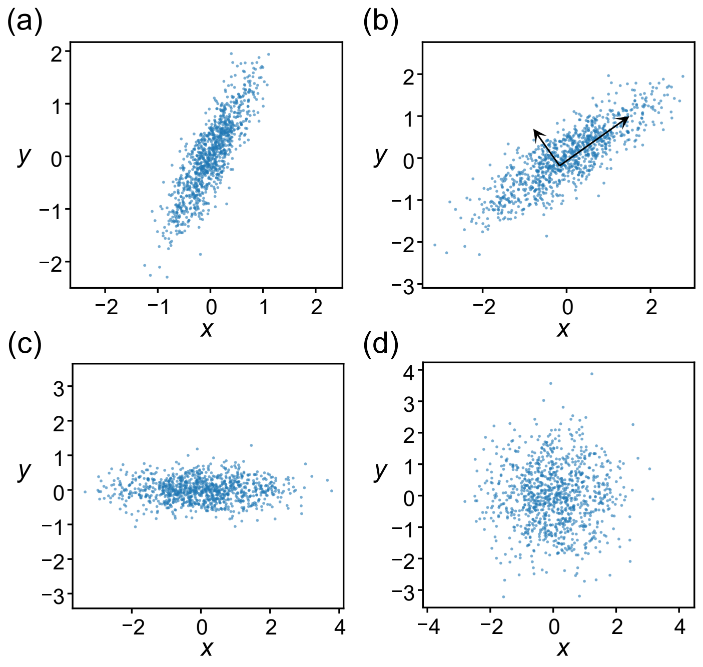


Fig. 2.3: Schematic illustration of input preprocessing process. (a) is the raw data distribution, (b) is scaled data, (c) is PCA-transformed data, and (d) is whitened data. First, each component in the raw data is scaled to a reasonable range. Then, PCA transformation is applied to the scaled data. Finally, whitening can be applied to the decorrelated data to scale each principal components.

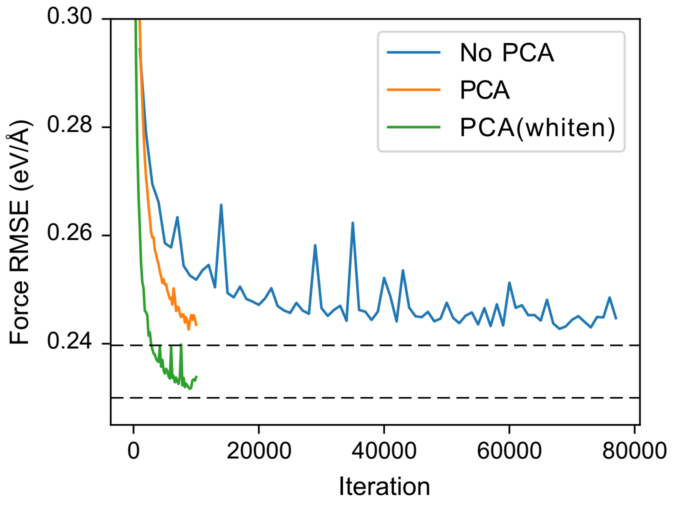


Fig. 2.4: Convergence of validation force RMSE against the training iteration with and without PCA and whitening. The blue line indicates the results without PCA preprocessing, and the orange line is the results with PCA preprocessing, but no whitening. The green line shows the results when both PCA and whitening are applied. The horizontal dashed lines indicate the converged RMSE value without PCA and with PCA and whitening. The training set consists of molecular dynamics trajectories of liquids and crystals in GeTe system.

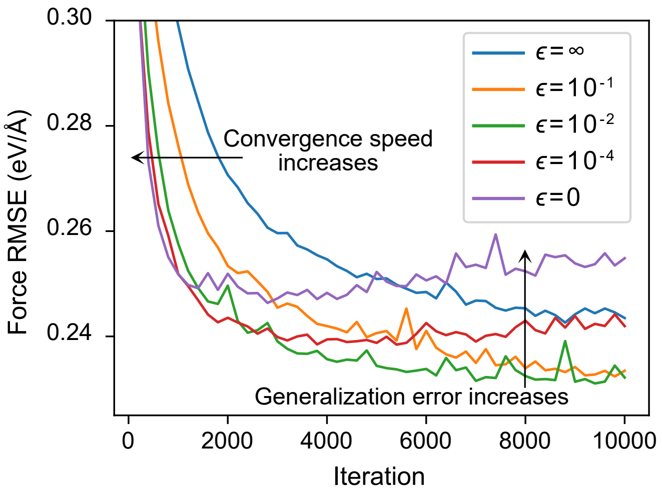


Fig. 2.5: Convergence of validation force RMSE against the training iteration with varying constant ϵ in eqn 2.39. $\epsilon = 0$ corresponds to conventional whitening, and $\epsilon = \infty$ indicates PCA without whitening. The RMSEs for the training set are not shown, but they are in the similar range regardless of the value of ϵ (0.22–$0.23\,\mathrm{eV/Å}$). The trade-off between convergence speed and generalization error is clearly observed where both convergence speed and generalization error decreases as ϵ increases. The training set consists of molecular dynamics trajectories of liquids and crystals in GeTe system.

the learning with gradient descent in the shallow neural network (linear regression) can be modeled as dynamic equations. Therefore, one can formulate generalization error E_g as a function of time. (In the shallow neural network, each mode is independent, but in the deep network, there is a coupling between modes)

$$E_g\left(t\right) = \frac{1}{N}\sum_i \left[\left(\sigma_w^2 + \left(\sigma_w^0\right)^2\right) e^{-\frac{2\lambda_i t}{\tau}} + \frac{\sigma_\epsilon^2}{\lambda_i}\left(1 - e^{-\frac{\lambda_i t}{\tau}}\right)^2\right] + \sigma_\epsilon^2\,, \qquad (2.40)$$

where i is the index of a mode (principal component), λ_i is the eigenvalue (variance) of an ith mode, and σ_ϵ is the error in the output.

Although the eqn 2.40 is derived from a shallow linear network, it can be seen that the principal component with smaller variance gives more serious generalization error, which agrees well with our observations. The use of eqn 2.39 would not be the optimal scaling, but we found that it gives nice results with much faster convergence and lower generalization error than eqn 2.38 with ϵ tuned as a hyperparameter.

2.2.4 Training

Training neural network is a process of minimizing error by updating weights, which sounds simple enough. However, in practice, there are much more things to be considered. In this section, we describe some important aspects of training neural network potentials.

Training set

The training set is a data set to which a machine learning model is trained. It consists of descriptors for each atom and the reference energy (optionally atomic forces and virial stresses) for selected snapshots. It is a heart of machine learning potential. The accuracy of a machine learning potential heavily relies on the training set. Furthermore, the training set determines the reliability and applicability of machine learning potential. In order to describe a certain system, a training set must cover all local configurations that can appear in the system because machine learning models cannot extrapolate well, and they are not based on any physical rule. If the local environment is not covered by the training set, machine learning potential can give totally unreasonable results, leading to a catastrophic failure of a simulation.

The training set is usually constructed from the trajectory of molecular dynamics simulation and some distorted crystal structures. (Other sampling methods are discussed in Chapter 4) The molecular dynamics simulation is often carried out at a higher temperature than the target temperature to increase the stability of potentials. High-temperature molecular dynamics is also beneficial as distinct configurations would be more *connected* at the higher temperature (see the discussion in Chapter 3) There is also a problem of sampling bias inherent to the modeling, which is discussed below.

Molecular dynamics simulations are often performed under the convergence condition that might not be sufficient for the training. The errors in the reference data would affect the training negatively. Therefore, it is advised to test convergence and perform oneshot calculations on the selected snapshots to collect more accurate data.

Optimization

When the training set is given, the error between the reference and the prediction is a function of weights (and biases) of the network. The loss function, which is minimized during the training process, is defined as the error between the reference and the prediction. The training process is basically the optimization of loss function with respect to the weights. Therefore, it is important to set proper functional form for the loss function. The common functional forms of loss functions for energy, force, and virial stress are defined as:

$$\Gamma_E = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{E_i^{\mathrm{DFT}} - E_i^{\mathrm{NNP}}}{N_i}\right)^2 , \tag{2.41}$$

$$\Gamma'_E = \frac{1}{M}\sum_{i=1}^{M}\left(E_i^{\mathrm{DFT}} - E_i^{\mathrm{NNP}}\right)^2 , \tag{2.42}$$

$$\Gamma_F = \frac{1}{3\sum_{i=1}^{M} N_i}\sum_{i=1}^{M}\sum_{j=1}^{N_i}\left|\mathbf{F}_{ij}^{\mathrm{DFT}} - \mathbf{F}_{ij}^{\mathrm{NNP}}\right|^2 , \tag{2.43}$$

$$\Gamma'_F = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{3N_i}\sum_{j=1}^{N_i}\left|\mathbf{F}_{ij}^{\mathrm{DFT}} - \mathbf{F}_{ij}^{\mathrm{NNP}}\right|^2 , \tag{2.44}$$

$$\Gamma_S = \frac{1}{6M}\sum_{i=1}^{M}\left|\mathbf{S}_i^{\mathrm{DFT}} - \mathbf{S}_i^{\mathrm{NNP}}\right|^2 , \tag{2.45}$$

where M is the total number of structures in the training set, and N_i is the number of atoms in an ith structure. E_i, $\mathbf{F}_{ij}$, and $\mathbf{S}_i$ are the energy of the ith structure, the force of a jth atom in the ith structure, and the virial stress of the ith structure, respectively. There is yet no consensus on the functional form of loss functions. Two different equations for energy loss and force loss (and their combinations) are commonly used throughout the community.[30, 42–47] (some adopt exponential loss functions to reduce outliers[28]) The $1/M$ and $1/(3\sum N_i)$ terms average the loss function over structures and atoms, respectively. It is both accepted for those terms to be included or omitted.

The difference between eqn. 2.41 and eqn. 2.42 is that eqn. 2.41 divides the energy error by the number of atoms while eqn. 2.42 does not. When the average energy error is larger than the variance of error, i.e., there is a constant energy shift, the energy error is proportional to the number of atoms in the structure (the error would be proportional to $\sqrt{N_i}$ when the average error becomes smaller than the variance). In such a case, one can treat every energy quantity equivalently by dividing the error by the number of atoms (eqn. 2.41). On the other hand, the definition in eqn. 2.42 would make the training biased toward the structure with more atoms. In real applications with the complex training set, the average energy error does reach zero for each group of structures due to intrinsic errors from the approximations (see below). Therefore, we use eqn. 2.41 for energy loss. The constant $1/M$ normalizes the loss function. We include the normalization constant as it is more convenient to have a loss function whose magnitude is independent of the size of the training set. Otherwise, one has to adjust a learning rate or a regularization constant according to the size of the training set. Since μ and ν can balance the magnitudes between loss terms (see below), the normalization constant is just a matter of convenience.

The difference between eqn. 2.43 and eqn. 2.44 is that eqn. 2.44 averages the square of force error for each structure. This makes all snapshots contribute equally regardless of the number of atoms. That means the force in the large structure has a smaller contribution than the force in the small structure. We expect each force quantity to contribute equally, so we use eqn. 2.43 for force loss. The normalization constant $1/(3\sum N_i)$ is included for convenience. The loss function for virial stress is defined similarly to the force loss.

With all ingredients, the total loss function is simply defined as a sum of three losses (energy, force, and stress) with the coefficients μ and ν to balance units and the magnitude between energy, force, and stress:

$$\Gamma = \Gamma_E + \mu\Gamma_F + \nu\Gamma_S\,. \tag{2.46}$$

The coefficients μ and ν can be set to equalize the magnitude of three loss terms so that three loss components contribute equally to the minimization. For specific training procedures, one can adjust coefficients to achieve focused training on either energy, force, or stress.

Before we start the optimization process, the weights are first initialized randomly to follow a certain distribution, usually (truncated) normal or uniform distribution. The range of initial weight distribution is important. If weights are initialized either too large or too small, then the optimization process can be severely hindered. Also, too large initial weights can result in the model with high generalization error. We used He initialization[48], which adjusts the range of node values to stay in the reasonable range, taking the number of nodes in the previous layer into account.

In a simple gradient descent method, each weight is updated by the gradient of loss function multiplied by a learning rate α:

$$w \to w - \alpha\frac{\partial\Gamma}{\partial w}\,. \tag{2.47}$$

The learning rate α is an important hyperparameter. If the learning rate is too large, then the converged error remains high or even completely fail to optimize. In contrast, if the learning rate is too low, the convergence becomes too slow. Therefore, one should find a balanced learning rate. We employed exponentially decaying learning rate to accelerate the training early and improve the convergence later.

Instead of updating weights once per the entire training set (full-batch), we update weights per mini-batch of the training set, where mini-batch contains a fixed number of snapshots (only a small fraction of the entire training set). As one shuffles the training set and split into mini-batches, it is called stochastic gradient descent (SGD). SGD requires less memory and more efficient than a standard gradient descent method. SGD is widely used in machine learning, as it shows some desirable properties. Owing to the stochastic nature of SGD, it is also known to help avoiding bad local minima and reducing generalization error.

A standard gradient descent method gives slow convergence. Therefore, advanced optimizers such as AdaGrad[49] and Adam[50] were suggested, which adjust the learning rate for each parameter based on the momentum. Other optimizers such as L-BFGS[44, 47], global extended Kalman filter[51], and Levenberg-Marquardt algorithm[44] were also used to train neural network potentials. In this dissertation, we use Adam optimizer, as we found it efficient.

Note that unlike conventional machine learning problem, we train our network by minimizing the error of a sum of outputs (total energy is a sum of atomic energies). In the conventional *ab initio* calculations, atomic energy is not provided, so we cannot train with atomic energies. (We note a study that trained atomic NN directly with DFT atomic energies[52]) As we are training forces, which is atomic quantity, one might think we are giving direct information on each atom. However, atomic forces are also the (weighted) sum of derivatives of atomic energy ($\mathbf{F}_i = -\sum_j \partial E_{\text{at},j}/\partial \mathbf{G}_j \times \partial \mathbf{G}_j/\partial \mathbf{r}_i$). The implication of training over the sum is discussed in Section 3.

Overfitting

A model can be trained to give accurate results only to the training set and cannot predict the structures that is similar enough to the training set, but not directly included in the training set. This means the trained model is specialized toward the training set and gives high generalization error. This is called overfitting.

Overfitting should be prevented as such overfitted models would be inappropriate for simulations. Overfitting can be severe when the model capacity (the size of the network) far exceeds the number of training data. Therefore, it is first advised to either collect more data or adjust the size of the network so that the network gives sufficiently small error without overfitting.

One can apply explicit regularization techniques such as weight decay and dropout to prevent overfitting. The weight decay, or L_2 regularization, is a method that adds penalty term proportional to the L_2 norm of weights to the loss function.

$$\Gamma_{L_2} = \lambda \left\| w \right\|_2^2 . \tag{2.48}$$

The norm of weights has to be large to have overfitting. Therefore, giving a penalty to the norm of weights is one simple way to prevent overfitting. However, one should be careful about the choice of a regularization constant λ as too large λ would result in high prediction error.

Dropout[53] is a technique that reduces overfitting by preventing nodes from co-adapting too much. Overfitting occurs when all weights are tuned to work together on the training set. By randomly *turning off* some of the nodes (and weights) during each iteration, one introduces stochasticity (noise) into the parameter update. It is believed to prevent co-adaptation on the training data and overfitting. It can also be viewed as an efficient method to perform model averaging.

Implicit regularization includes SGD and early stopping. It is known that SGD also works as regularization, resulting in the minima with the small norm.[54] In the early stopping method, one prepares a *validation set*, which is different from the training set and the test set. Usually, the whole data set is split into a training set, a validation set, and a test set. However, if the sampling rate from dynamics trajectory (the time interval between snapshots) is too low, the validation set cannot validate the model as the errors between sequential snapshots are correlated. In such a case, it is recommended to generate a validation set from an independent trajectory from the training set. Then, one monitors the error of the training set and validation set during the training process. The validation error increases when the overfitting occurs. We stop training early just before the validation error increases. It is recommended to always monitor the validation error to detect any sign of overfitting.

Noise in data

One might think that the data used in machine learning potential is noiseless since the data is computed from well-defined equations (e.g., symmetry functions and *ab initio* calculation results). However, several factors introduce errors into the data.

First, the most obvious one is the numerical error from *ab initio* calculations. For example, plane-wave based density functional theory calculation uses a plane-wave basis set with certain cutoff energy and k-point grid. By using a finite basis set and k-points, a numerical error is introduced. Additionally, the electronic step might not have fully converged.

Second, the high-dimensional local environment (dimension of $3N_{\text{neighbor}} - 6$) is encoded into a symmetry function vector, which has a much lower dimension. This means that two distinct local environments, which have different atomic energies, can have similar symmetry function vectors. Therefore, even if *ab initio* calculation has no error, it acts as a noise. In other words, hidden dimensions in input effectively introduces noise in the output.

Finally, an error is introduced from the use of finite cutoff (transferability issue). This is similar to the second source of error. The transferable range of atomic energy in *ab initio* calculations can be larger than the finite cutoff we use (usually 6–7 Å). In that case, the movement of atoms outside of the cutoff range changes the atomic energy, but the symmetry function with the finite cutoff cannot see the change. Therefore, hidden energy contribution becomes equivalent to the noise in output data.

Sampling bias

Intrinsic sampling bias is introduced to *ab initio* data as a whole structure is needed to model an atom with specific environment. That is, one has to model the specific atom with other *background* atoms. For instance, when modeling a defect, one constructs a large supercell that incorporates one defect and many crystalline atoms. Therefore, the training set intrinsically includes much more crystalline atoms than the defective atoms. This imbalance can steer machine learning potential to be trained specifically toward crystalline atoms, resulting in much higher error for defective atoms that are of our interest.

This kind of sampling bias is inherent to *ab initio* calculation and cannot be solved by simply adding more data; when one adds more data, one is adding more background atoms, too. The problem is that those minority atoms, such as defects, are what we are interested in. It was demonstrated in the Ref. [13] that the higher error of defective atoms can cause catastrophic failures in the simulation. Therefore, it would not be ideal to treat all force quantities equally, as eqn 2.43. In such a case, training with more weight on the sparsely sampled atoms of interest would be helpful. Jeong et al. proposed a weighting scheme utilizing Gaussian density function (GDF) to deal with the sampling bias inherent to *ab initio* data and improve the reliability of machine learning potential.[13] They weighted atomic forces by the function of the Gaussian density of each atom. The modified loss function for force is defined as:

$$\Gamma_F = \frac{1}{3\sum_{i=1}^{M} N_i} \sum_{i=1}^{M} \sum_{j=1}^{N_i} \Theta\left[\rho\left(\mathbf{G}_{ij}\right)^{-1}\right] \left|\mathbf{F}_{ij}^{\mathrm{DFT}} - \mathbf{F}_{ij}^{\mathrm{NNP}}\right|^2 , \tag{2.49}$$

where $\mathbf{G}_{ij}$ is the symmetry function vector of jth atom in ith snapshot, ρ is the Gaussian density function, and Θ is a weighting function. Although atomic force is strictly not a local quantity that is determined by a symmetry function vector, it was demonstrated to reduce force error on sparsely sampled atoms, give better description of high energy barriers, and make the simulation more reliable.

2.3 Metadynamics

Molecular dynamics (MD) is a standard tool to study the atomistic motion and understand the underlying mechanisms in various fields of science. However, when multiple energy minima are separated by high free energy barriers or when the system is governed by a slow diffusion process, it can be nearly impossible to sample all energetically relevant configurations in a feasible computational time. If one wants to sample configuration space, the sampling is inefficient as the kinetic bottleneck hinders the sampling. Also, obtaining sufficient sampling of rare events, in which we are often interested, is extremely difficult. In order to improve the sampling of molecular dynamics, advanced sampling methods, such as metadynamics[55], umbrella sampling[56], and adaptive force bias[57], have been suggested.

Metadynamics[55] is a powerful simulation technique that enhances sampling, where history-dependent bias potential acts on collective variables. Collective variables (CVs) are the function of atomic coordinates that can distinguish all metastable states of interest. Since the space to explore increases exponentially as the number of CVs grows, it is also desirable to keep a small set of CVs. By adding bias potential to the original potential energy, we are adding biasing force so that the system evolves into less explored states. It can be pictured as filling sand into the minima so that the system escape the local minima and explore various configurations (see Fig. 2.6). The bias potential is defined as a Gaussian form:

$$U_b\left(\mathrm{s}\right) = \sum_i h\mathrm{exp}\left(-\frac{\left|\mathrm{s}-\mathrm{s}_i\right|^2}{2\sigma^2}\right), \tag{2.50}$$

where $\vec{s}$ is the collective variable vector, and the summation is over CV of all visited configurations that are added in the interval of Δt during the metadynamics simulation. h and σ controls the height and the width of the bias potential, respectively.

The additional biasing force due to the bias potential in the real space can be computed as:

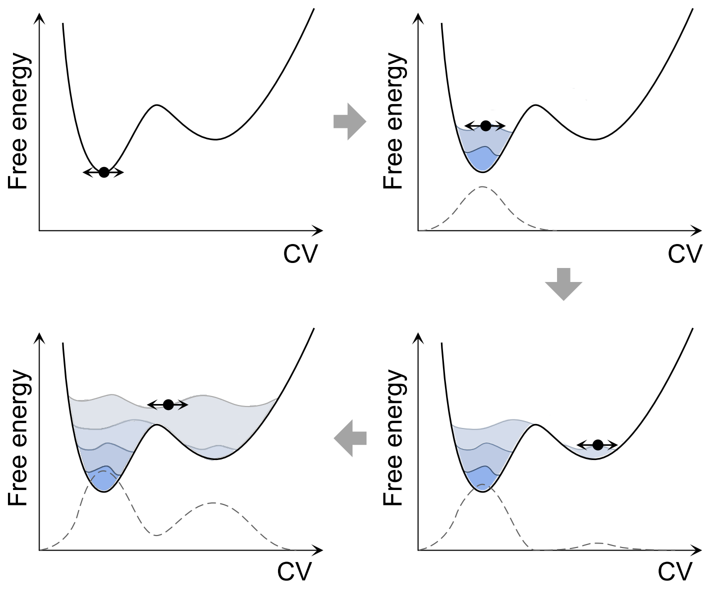


Fig. 2.6: Schematic illustration of the metadynamics method. At first, the system is trapped in a local minimum. By adding bias potential, the system can escape the local minimum and explore other minima. The energy surface becomes flat after sufficient time.

$$F_{i,\alpha} = -\frac{\partial U_b}{\partial R_{i,\alpha}} = -\sum_{j=1}^{N_{\text{CV}}} \frac{\partial U_b}{\partial s_j} \frac{\partial s_j}{\partial R_{i,\alpha}}\,, \tag{2.51}$$

where s_j is jth collective variable.

One of the advantages metadynamics provides is that one can reconstruct the free energy surface from the bias potential. In order to obtain well converged free energy surface, well-tempered metadynamics[58], which reduces the Gaussian height h gradually, is used to increase the convergence. However, we do not delve into the details about the relation between the bias potential and the free energy surface as we do not use metadynamics to compute the free energy surface.

Many variants of metadynamics were proposed,[58–61] but they are not covered in the dissertation. In the following section, we describe the adaptive Gaussians method, which was applied to our simulation.

2.3.1 Adaptive Gaussians

Equation 2.50 is defined with the assumption that collective variables are independent. Using eqn 2.50 would make exploration difficult or inefficient when collective variables are highly correlated. Equation 2.50 can be modified to use generic multivariate Gaussian function as:

$$U_b(\mathbf{s}) = \sum_{i=1}^{N_{\text{sample}}} h \exp\left[-\frac{1}{2}\sum_{j=1}^{N_{CV}}\sum_{k=1}^{N_{CV}} \left(s_j - s_j^{(i)}\right) \sigma_{jk}^{-2,(i)} \left(s_k - s_k^{(i)}\right)\right], \tag{2.52}$$

where $s_j^{(i)}$ is jth collective variable of ith visited sample and σ_{jk} determines the shape of Gaussian potential.

The determination of σ is not simple as it is not a fixed value, but depends on the value of s. Branduardi et al. suggested adaptive Gaussians, which determine the optimal shape of Gaussian potential on-the-fly during the simulation.[62] σ_{ij}^2 is proportional to the covariance between two collective variables s_i and s_j.

$$\sigma_{ij}^2 \propto \langle \Delta s_i \Delta s_j \rangle \tag{2.53}$$

Branduardi suggested two methods of estimating the covariance matrix between the collective variables. One is time-dependent covariance, and the other is position-dependent (or geometry-dependent) covariance.

The time-dependent covariance estimation is named dynamically-adapted Gaussians. Simply put, the covariance and the center are estimated as a time average from the last part of the simulation trajectory. They introduced an exponential weighting function with decay time τ_D so that the center and covariance at time t are given by:

$$\bar{s_i}(t) = \frac{1}{\tau_D} \int_0^t dt' s_i\left(t'\right) e^{-(t-t')/\tau_D} , \tag{2.54}$$

$$\sigma_{ij}^2(t) = \frac{1}{\tau_D} \int_0^t dt' \left[s_i\left(t'\right) - \bar{s_i}\left(t'\right)\right] \left[s_j\left(t'\right) - \bar{s_j}\left(t'\right)\right] e^{-(t-t')/\tau_D} . \tag{2.55}$$

Then $\bar{s_i}$ and σ_{ij}^2 are evolved during the simulation using the derivatives of eqn 2.54 and eqn 2.55. The downside of the dynamically adapted Gaussian is that it requires averaging over a certain period of time to get a reliable estimation. Therefore, it might be inappropriate to be applied to a fast-evolving system. Furthermore, since we carry out metadynamics simulations with DFT calculations, such time averaging is too time-consuming.

The position-dependent covariance does not rely on time average but imposes additional assumptions to evaluate the covariance. First, it is assumed that the change of CV can be linearly approximated as:

$$\Delta s_i \approx \sum_\alpha \frac{\partial s_i}{\partial q_\alpha} \Delta q_\alpha , \tag{2.56}$$

where q_α is αth microscopic coordinate. We assume that q_α is an independent variable from the distribution $\mathcal{N}(0, \sigma_G^2)$ so that $\langle q_\alpha q_\beta \rangle = \delta_{\alpha\beta}\sigma_G^2$. Then:

$$\sigma_{ij}^2(q) = \langle \Delta s_i \Delta s_j \rangle \approx \sigma_G^2 \sum_\alpha \frac{\partial s_i}{\partial q_\alpha} \frac{\partial s_j}{\partial q_\alpha} \, . \tag{2.57}$$

Equation 2.57 depends only on the microscopic variable q. We adopted geometry-adapted Gaussians in our simulation as it can be evaluated instantaneously without a time average. The Ref. [62] also covers the free energy estimation from modified bias potential, but we do not cover free energy aspects in this dissertation.

From a computation point of view, the use of the full covariance matrix requires $N_G(N_G + 3)/2$ floating points per sample (cf. N_G floating points for isotropic covariance). The computational cost increases as the simulation continues. It can be quickly prohibiting. However, we found that the cost of DFT calculation is much higher than the cost of bias evaluation with full covariance, so it is not a problem in our applications.

If one finds the cost of bias evaluation becoming prohibiting, one can maintain the cost of barrier evaluation by approximating bias potential with a smaller number of Gaussian functions. Many approximation methods were suggested: VKL[63], hierarchical EM (HEM)[64], L2U[65], density-preserving hierarchical EM algorithm (DPHEM)[66].

We note that the inverse of the covariance matrix can be numerically unstable when the correlation between collective variables is too high. We regularize the covariance matrix so that the condition number remains at the certain value. In that case, the sampling might not be enhanced along with the principal components with very small variance.

Chapter 3

Atomic energy mapping

3.1 Introduction

Most machine learning potentials, including high-dimensional neural network potential, are based on the assumption that the total energy of a system is a sum of transferable atomic energies. *Transferable* atomic energy means that the atomic energy is determined by only atomic arrangement within a certain cutoff distance so that atomic energy is identical, given the local environment within the cutoff radius is identical.

Machine learning potentials learn atomic energies from the total energy of density functional theory (DFT) calculations. Due to this unconventional machine learning structure, two issues arise. First, transferable atomic energy must be defined within the DFT level for machine learning potential to be established. Second, the accurate total energies for the training set does not necessarily guarantee the accuracy in atomic energy, which is essential for transferability. However, to our knowledge, such issues have never been discussed explicitly.

In this chapter, we first show that transferable atomic energy is defined within DFT, especially paying attention to the transferable range. Then through some examples, we show that NNP is capable of learning atomic energies from total energy but also prone to learning incorrect atomic energy function.

3.2 Transferable atomic energy within DFT

Machine learning potentials are based on the representability of DFT total energy as a sum of transferable atomic energies that depend on the atomic arrangement within a cutoff radius R_c:

$$E_{\text{tot}}^{\text{DFT}} = \sum_i E_{\text{at}} \left(\mathcal{R}_i ; R_c \right) \ , \tag{3.1}$$

where i is atom index and $\mathcal{R}_i$ is the relative position vectors within R_c. The concept of locality in a matter is widely accepted throughout the various fields such as physics, chemistry, and materials science.[24] It is also the basis of classical potentials. If transferable atomic energy cannot be defined in a DFT level, it would be impossible to train a transferable machine learning potential based on eqn 3.1 with DFT total energies. It is well known that the atomic energy can be defined by integrating DFT energy density on atomic volumes.[67, 68] However, the existence of atomic energy in DFT level does not guarantee that it is transferable to similar environments, which is essential to machine learning potentials. To our knowledge, it is not rigorously discussed elsewhere whether the transferable atomic energy can be defined within DFT or not. In this section, we first check whether eqn 3.1 is justified by explicitly deriving the definition of transferable atomic energy within DFT. We start with locality or *nearsightedness* of electronic structure[23, 24] and pay attention to the transferable range. In the following discussions, we assume a unary system that is large enough such that cutoff spheres do not self-overlap under the periodic boundary conditions, and wave functions are effectively real-valued for the simplicity.

Within the semilocal density approximation, $E_{\text{tot}}^{\text{DFT}}$ can be expressed in terms of electron density $\rho(\mathbf{r})$ and one-electron density matrix $\rho(\mathbf{r}, \mathbf{r}')$ (eqn 2.13) as a sum of kinetic energy, exchange correlation energy, and Coulomb energy:

$$
\begin{aligned}
E_{\text{tot}}^{\text{DFT}} &= E_{\text{kin}} + E_{\text{XC}} + E_{\text{Coul}} \\
&= -\frac{1}{2}\int \nabla_{\mathbf{r}}^2 \rho(\mathbf{r},\mathbf{r}')|_{\mathbf{r}=\mathbf{r}'} d\mathbf{r}' + \int \rho(\mathbf{r})\varepsilon_{\text{XC}}(\rho(\mathbf{r}),\nabla\rho(\mathbf{r}))d\mathbf{r} \\
&+ \frac{1}{2}\int \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}d\mathbf{r}' - \sum_i \int \frac{q_i\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{r}_i|}d\mathbf{r} + \sum_{i>j}\frac{q_i q_j}{|\mathbf{r}_i-\mathbf{r}_j|},
\end{aligned}
\tag{3.2}
$$

where q_i and $\mathbf{r}_i$ are the charge and position of the ith ion. Under the assumption that $\mathcal{O}(N)$ methods[26, 27] work well for given systems, we explicitly show that each energy term can be split into atomic contributions that are defined by local atomic arrangement within a certain cutoff radius, and therefore transferable atomic energy can be defined within DFT.

We start by partitioning the volume into non-overlapping atomic volumes such as Voronoi cell. Let V_i be the volume containing ith atom. We define a charge density inside atomic volume: $\rho_i(\mathbf{r}) = \rho(\mathbf{r})[\mathbf{r} \in V_i]$, where the squared bracket is the Iverson bracket whose value is 1 when the logical proposition inside is true, and 0 otherwise. The atomic exchange correlation energy can be simply defined by substituting $\rho_i(\mathbf{r})$ to $\rho(\mathbf{r})$ in the integrand of E_{XC}:

$$
E_{\text{XC},i} = \int \rho_i(\mathbf{r})\varepsilon_{\text{XC}}(\rho_i(\mathbf{r}),\nabla\rho_i(\mathbf{r}))d\mathbf{r} \tag{3.3}
$$

It is obvious that $E_{\text{XC}} = \sum_i E_{\text{XC},i}$ as $\rho(\mathbf{r}) = \sum_i \rho_i(\mathbf{r})$. The nearsightedness principles states that the perturbation of potential at a certain point does not affect the charge density at the other point if two points are far enough, given the local chemical potential of electrons is fixed.[23, 24] This means that $\rho_i(\mathbf{r})$, and hence $E_{\text{XC},i}$, is determined by the atomic arrangements within a certain cutoff radius (R_{c}^1) from $\mathbf{r}_i$.

Next, we define the total charge density in the volume V_i: $\rho_{\text{tot},i}(\mathbf{r}) = q_i\delta(\mathbf{r}-\mathbf{r}_i) - \rho_i(\mathbf{r})$. Then, we can define atomic Coulomb energy, $E_{\text{Coul},i}$, whose sum equals E_{Coul}:

$$\begin{aligned} E_{\text{Coul},i} &= \frac{1}{2}\sum_{j\neq i}\int \frac{\rho_{\text{tot},i}(\mathbf{r})\rho_{\text{tot},j}(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}d\mathbf{r}' \\ &+ \frac{1}{2}\int \frac{\rho_i(\mathbf{r})\rho_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}d\mathbf{r}' - \int \frac{q_i\rho_i(\mathbf{r})}{|\mathbf{r}-\mathbf{r}_i|}d\mathbf{r}\,, \end{aligned} \tag{3.4}$$

where the first term is the energy between electrons in V_i and V_j, the second term is electron-electron interaction energy within V_i, and the last term is the interaction energy between ith ion and electrons within V_i. The second and the third term is locally defined within V_i. However, the first term is long-ranged and cannot be described with the finite cutoff. Here, we assume the system where the electrostatic interaction is negligible beyond a certain cutoff radius (R_c^2) due to cancellation. It is a reasonable assumption for condensed systems with weak ionic characters. The success of short-ranged NNP for many condensed systems, such as SiO_2[47] and GeTe[69], supports the assumption. For the systems where long-range interaction becomes important, the long-range Coulomb interaction can be described separately with atomic point charges, which are predicted through separate atomic neural networks.[70, 71] Therefore, we neglect the Coulomb interaction between $\rho_{\text{tot},i}$ and $\rho_{\text{tot},j}$ if $|\mathbf{r}_j - \mathbf{r}_i| > R_\text{c}^2$. Then, eqn 3.4 is locally defined by the atoms within $R_\text{c}^1 + R_\text{c}^2$ because $\rho_i(\mathbf{r})$ is determined by the atoms within R_c^1 (neglecting the size of V_i).

As the last step, we define local atomic kinetic energy. Since $\rho(\mathbf{r},\mathbf{r}')$ decays exponentially as $|\mathbf{r}-\mathbf{r}'|$ increases in metals at finite temperature or insulators,[24, 25] $\rho(\mathbf{r},\mathbf{r}')$ can be neglected if $|\mathbf{r}-\mathbf{r}'|$ is larger than some threshold (R_c^3) as utilized in the divide-and-conquer method[27]. Therefore, $\rho(\mathbf{r},\mathbf{r}')$ at $\mathbf{r}$ is determined by the atomic arrangement within R_c^4, which is larger than R_c^3. We define projected density matrix $\rho_{ij}(\mathbf{r},\mathbf{r}') = \rho(\mathbf{r},\mathbf{r}')[\mathbf{r} \in V_i][\mathbf{r}' \in V_j]$. Then, the atomic density matrix $\rho_{\text{at},i}(\mathbf{r},\mathbf{r}')$ is defined as follows:

$$\rho_{\text{at},i}(\mathbf{r},\mathbf{r}') = \rho_{ii}(\mathbf{r},\mathbf{r}') + \frac{1}{2}\sum_{j\neq i}^{|\mathbf{r}_j-\mathbf{r}_i|<R_\text{c}^3} \rho_{ij}(\mathbf{r},\mathbf{r}')\,. \tag{3.5}$$

The atomic density matrix satisfies $\rho(\mathbf{r},\mathbf{r}') = \sum_i \rho_{\text{at},i}(\mathbf{r},\mathbf{r}')$ and depends on the atoms within R_c^4 from ith atom (again, neglecting the size of V_i). The atomic kinetic energy is then defined as the following:

$$E_{\text{kin},i} = -\frac{1}{2}\int \nabla_{\mathbf{r}}^2 \rho_{\text{at},i}(\mathbf{r},\mathbf{r}')|_{\mathbf{r}=\mathbf{r}'} d\mathbf{r}' \,. \tag{3.6}$$

The sum of the atomic kinetic energy equals the total kinetic energy as the kinetic energy operator is linear.

Combining individual atomic energy terms we defined above, the atomic energy of ith atom can be defined within DFT calculations:

$$E_{\text{at},i} = E_{\text{kin},i} + E_{\text{XC},i} + E_{\text{Coul},i} \,, \tag{3.7}$$

where $E_\text{tot}^\text{DFT} = \sum_i E_{\text{at},i}$ and $E_{\text{at},i}$ is determined by the atomic arrangement within the cutoff distance of $R_\text{c} = \max(R_\text{c}^1 + R_\text{c}^2, R_\text{c}^4)$. Note that the atomic energy defined here is not unique because it depends on the way to define the atomic volume.

The existence of transferable atomic energy in DFT implies that the objective of the machine learning potential is to identify the underlying atomic energy function when only their sums (total energies) are informed. This view is different from conventional views that NNP is an interpolation of total energies.[45, 72]

In order to obtain the atomic energy function in a computationally feasible way, two approximations are made. First, the cutoff distance is reduced from R_c, which should be fairly large for high accuracy, to a much smaller value of r_c, typically around 6–7 Å. This is a reasonable value for many systems as the chemical influence typically diminished rapidly. Second, the dimension of the local environment $\mathcal{R}$ is significantly reduced as we encode $\mathcal{R}$ to a descriptor vector. For neural network potentials with symmetry function vector $\mathbf{G}$, the approximation is written as:

$$E_\text{tot}^\text{DFT} = \sum_i E_\text{at}^\text{DFT}(\mathcal{R}_i; R_\text{c}) \simeq \sum_i E_\text{at}^\text{NN}(\mathbf{G}_i; r_\text{c}) \,. \tag{3.8}$$

The quality of NNP depends on how well $E_{\text{at}}^{\text{NN}}$ captures $E_{\text{at}}^{\text{DFT}}$. During the training process, the quality of NNP is usually assessed by root-mean-squared-error (RMSE) of total energy and forces. However, since we train NNP to the total energies, rather than directly to atomic energies, NNP can give accurate total energies for the training set, but the atomic energy function can deviate significantly from the reference DFT atomic energy function. We call it *ad hoc* energy mapping. (We note a recent effort to train NNP using atomic energies from DFT. [73]) In the following sections, we demonstrate that NNP is vulnerable to such ad hoc energy mapping, which undermines the transferability by giving wrong total energies for the structures outside of the training set. This can also leads to stability issues, which is discussed in Section 3.4. One thing to note is that the atomic force is given as a weighted sum of the atomic energy function ($F_{i,\alpha} = -\sum_j \partial E_{\text{at},j}/\partial G_{i,s} \cdot \partial G_{i,s}/\partial R_{i,\alpha}$). Therefore, although atomic forces are atomic properties, fitting atomic forces does not solve the problem of ad hoc energy mapping.

3.3 Examples of atomic energy mapping

Assessing the quality of atomic energy mapping is difficult as a direct comparison between $E_{\text{at}}^{\text{NN}}$ and $E_{\text{at}}^{\text{DFT}}$ cannot be made because $E_{\text{at}}^{\text{DFT}}$ is gauge-dependent. However, there are some gauge-invariant quantities we can compare. For instance, the atomic energy of diamond silicon is simply the total energy divided by the number of atoms because all atoms are equivalent. Another example is the surface energy. If we sum atomic energies over the surface region, it becomes well defined as gauge-dependent terms cancel out.[74] These invariant quantities allow us to examine the quality of the atomic energy mapping of NNPs.

In this section, we investigate the atomic energy mapping of NNP with four examples. First, we show that NNP is capable of inferring underlying atomic energy function by training NNP with the total energies of embedded atom method (EAM) where atomic energy is well defined so that atomic energies can be directly compared. The other examples, on the other hand, show that NNP is also vulnerable to ad hoc energy mapping by utilizing the invariant quantities. Specifically, the second and third examples are simple silicon crystal and slab model, respectively. They show different types of ad hoc energy mapping that arise from the limitations in the training set. The last example on silicon nanocluster is about the more practical situation, where monitoring the quality of NNP with only RMSE of total energies and forces does not reveal the quality of atomic energy mapping, possibly leading to NNP with the low quality of atomic energy mapping.

3.3.1 Classical potential

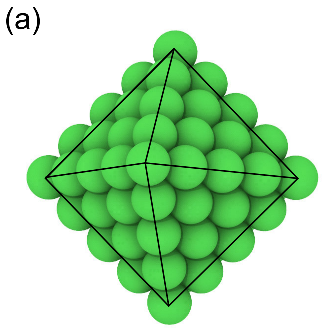
First, we examine whether NNP has the capability of identifying the underlying atomic energy function when only total energies are provided. We train NNP with the total energies of EAM potential[75], whose atomic energy is well defined as:

$$E_i = F\left(\sum_{j\neq i}\rho(r_{ij})\right) + \frac{1}{2}\sum_{j\neq i}\phi(r_{ij})\,, \tag{3.9}$$

where $F(\cdot)$ is the embedding function, ρ is a sum of pairwise electron densities, and ϕ is a pairwise potential. The atomic energy of EAM is virtually unique as it consists of only pairwise terms, making EAM a good candidate to compare atomic energies directly.

The training set consists of MD snapshots of Ni_{85} nanocluster at $500\,\mathrm{K}$. The structure of the nanocluster is shown in Fig. 3.1(a). After the training, RMSE is $0.3\,\mathrm{meV/atom}$ and $0.01\,\mathrm{eV/Å}$ for total energy and atomic forces, respectively.

Fig. 3.1(b) shows the correlation of atomic energies between EAM and NNP. A good correlation is found with atomic energy RMSE of $25\,\mathrm{meV}$. NNP successfully resolved different configurations such as corner, edge, surface, and bulk atoms. This example demonstrates the capability of NNP to identify the underlying atomic energy functions from total energies.



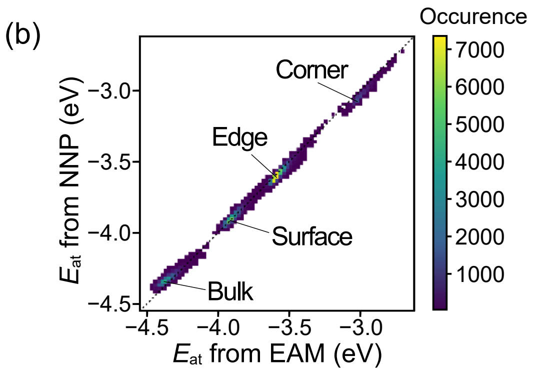


Fig. 3.1: (a) Structure of the relaxed Ni_{85} octahedron that consists of 6 corner, 36 edge, 24 surface, and 19 bulk atoms. (b) Two dimensional histogram plot that shows the correlation of atomic energy between EAM and NNP. The color indicates the density of point in each bin.

3.3.2 Si crystal example

In the following three examples, we train NNP to the total energies and forces of DFT calculations. The training and validation set consists of 350 and 150 MD snapshots of 64-atom silicon diamond crystal at $1000\,\mathrm{K}$, respectively. The MD is performed under NVT ensemble. After the training, RMSE is $1\,\mathrm{meV/atom}$ and $0.10\,\mathrm{eV/Å}$ for total energies and atomic forces, respectively. RMSEs for training and validation set are almost identical.

In this example, we investigate the atomic energy mapping by comparing the atomic energies along an equation of state (EOS), where atomic energies are uniquely defined. Due to the atomic vibration during MD, some atoms have a similar local environment to that of crystal under isotropic strain, which is a part of EOS. To show this clearly, we define $d_{\mathrm{NN}}(\mathbf{G})$ as the Euclidean distance in the symmetry function space to the nearest point in the training set (excluding self). Fig. 3.2(a) shows the distribution of d_{NN} for the training set and Fig. 3.2(b) shows the distribution of training points on the first two principal component axis (from PCA). The black solid circle shows the points which correspond to the structures of EOS. Fig. 3.2(c) shows d_{NN} for the same points in Fig. 3.2(b). As can be seen in Fig. 3.2(a), d_{NN} is less than 0.2 for most of the training set. The RMSE for the validation set, which is randomly selected from the training set, is almost identical to the training set. Therefore, we can regard a point to be included in the training set and therefore can be predicted well if $d_{\mathrm{NN}} < 0.2$. In Fig. 3.2(b) and (c), the square bracket indicates the range where d_{NN} is as low as the that of the training set (lower than 0.2), implying that EOS inside the square bracket could be learned although they do not belong to the training set. Therefore, if atomic energies are properly mapped, NNP should be able to predict the part of EOS inside the square bracket.

Fig. 3.2(d) shows the reference EOS (black dots) with the EOS predicted by NNP (blue line). The shade indicates the uncertainty by one standard deviation of predictions of five different NNPs. Interestingly, NNP predicts the energy at the equilibrium

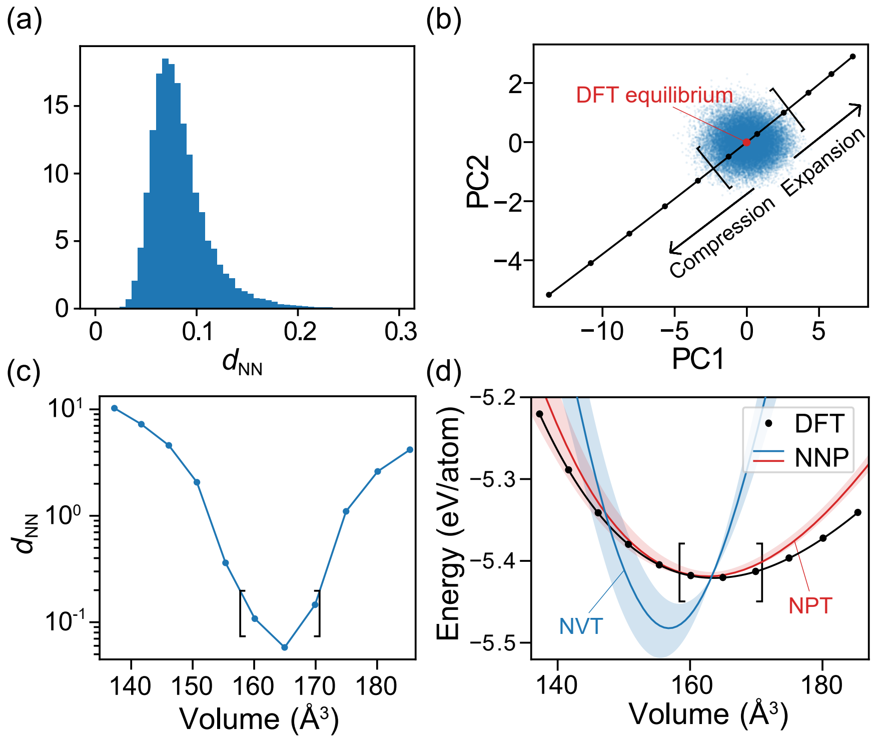


Fig. 3.2: (a) The distribution of d_{NN} for the training set. (b) The distribution of $\mathbf{G}$ in the training set (blue dots) and the equation of state (EOS) (black solid circles), projected onto principal component (PC) axes. (c) The d_{NN} for each point in EOS. In (b), (c), and (d), the square bracket indicates the same range for EOS where the $d_{\mathrm{NN}} < 0.2$. (d) The EOS of silicon diamond crystal for DFT and NNP. The blue and red solid lines are the average EOS over five NNPs that are trained with NVT and NPT MD snapshots, respectively. The shades are one standard deviation from the average, indicating the prediction uncertainty.

volume of DFT accurately but fails to predict the slope and other points near the equilibrium despite low RMSE for total energy. That is to say, NNP predicts the total energy accurately, but the atomic energies are markedly wrong, which corresponds to the ad hoc energy mapping.

The ad hoc energy mapping in this example happens because the training set consists of structures of a fixed volume. Locally, there is a volume expansion and compression. However, the expansion and compression occur at the same time, so they cancel out (although not exact). Consequently, any energy term in atomic energy that varies linearly to the volume does not change the total energy. Therefore, the slope of EOS at the equilibrium point becomes an arbitrary number. The ad hoc mapping can be removed by including structures with different volumes in the training set, or by including virial stress in the loss function. The red line in Fig. 3.2(d) shows EOS predicted with NNPs whose training set is constructed from MD snapshots under NPT ensemble. As the fluctuation of cell volume averts the cancellation of local volume changes, NNP can predict the slope and curvature of EOS reasonably.

3.3.3 Si slab example

In the third example of a silicon slab model, the training set consists of MD snapshots of Si(100)-(2$\times$2) slab with 128 atoms at a certain temperature between 100 and $1000\,\mathrm{K}$. The structure is shown in Fig. 3.3(a). Unlike the previous example of silicon crystal, we cannot directly compare individual atomic energies as they are not uniquely defined. Nevertheless, silicon atoms inside the bulk region have almost the same local environment to crystalline atoms (dashed circle indicate the cutoff sphere). Therefore, atomic energy in this region should be the same as that of crystalline atoms $[E_{\text{at}}^{\text{DFT}}(\text{bulk})]$. Since $E_{\text{tot}}^{\text{DFT}}$ is given, the average $E_{\text{at}}^{\text{DFT}}$ for the surface region (red atoms) can be obtained as $[E_{\text{tot}}^{\text{DFT}} - N_{\text{b}} \cdot E_{\text{at}}^{\text{DFT}}(\text{bulk})]/N_{\text{s}}$, where N_{b} and N_{s} are the number of atoms in the bulk and surface regions, respectively. By summing over the surface region, the value becomes independent of the atomic energy mapping.

Since the training set structures include the local environments of bulk and surface atoms, one would expect NNP to predict atomic energy well for both regions. Figure 3.3(b) shows the average atomic energy error $\Delta\bar{E}_{\text{at}}(\text{bulk})$ and $\Delta\bar{E}_{\text{at}}(\text{surface})$ for NNPs trained over MD snapshots of different temperatures. At the low temperature of $100\,\mathrm{K}$, while the RMSE of total energy is $0.3\,\mathrm{meV/atom}$, the average atomic energy error is -108 and $76\,\mathrm{meV/atom}$ for bulk and surface regions, respectively. This is another example of ad hoc energy mapping. NNP predicts total energies in the training set accurately, but the atomic energy mapping is wrong. The errors in bulk and surface regions cancel out since all snapshots in the training set consist of the same number of atoms in each region. In Fig. 3.3(b), it is noted that the atomic energy mapping error decreases gradually as the temperature of the training set increases. At the high temperature of $1000\,\mathrm{K}$, the magnitude of atomic energy error becomes comparable to RMSE of total energy ($3\,\mathrm{meV/atom}$).

To understand the temperature dependence of atomic energy mapping error, we examined the distribution of training points in the symmetry function space. Figure 3.3(c) shows the distribution of training points of 100 and $1000\,\mathrm{K}$, projected onto

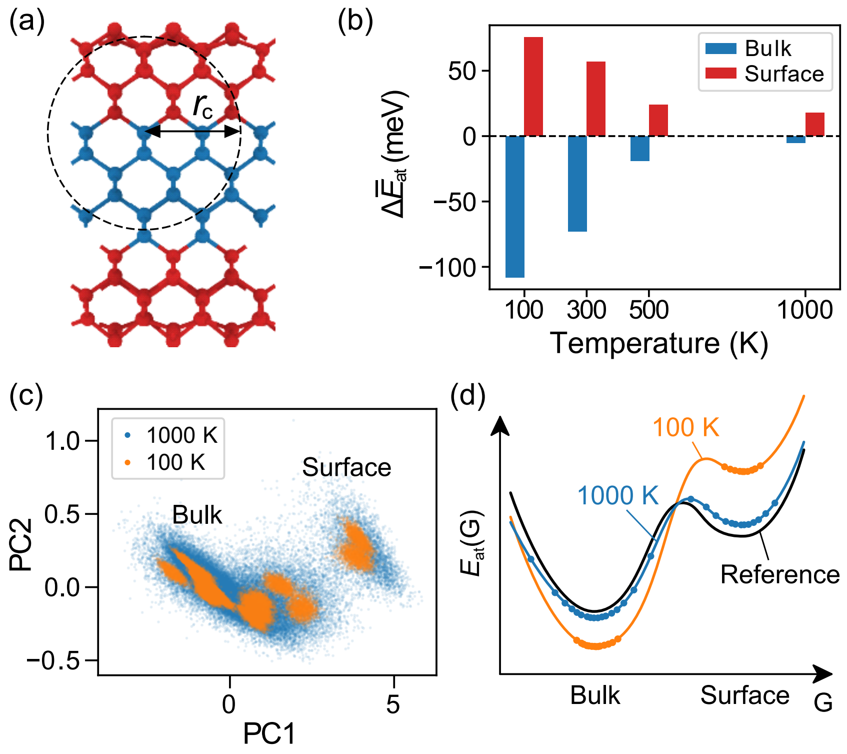


Fig. 3.3: (a) The structure of the relaxed Si(100)-(2×2) slab model. The atoms in bulk and surface regions are colored blue and red, respectively. r_c is the cutoff radius of symmetry functions. (b) The average atomic energy difference between DFT and NNPs for bulk and surface groups against the MD temperature of the training set. (c) Scatter plot of **G** in the training set, projected onto principal components (PC). (d) Schematic illustration of ad hoc energy mapping due to well-separated configurations.

the first two principal components. At $100\,\mathrm{K}$, the training points of bulk and surface regions are well separated. In contrast, training points at $1000\,\mathrm{K}$ have much broader distribution due to vibrations such that bulk and surface regions are more connected. As schematically drawn in Fig. 3.3(d), if two configurations are well-separated as in $100\,\mathrm{K}$, the offset between two configurations cannot be determined as any offset gives almost the same total energy. On the other hand, if two configurations are connected as in $1000\,\mathrm{K}$, the information on the intermediate configurations help to adjust the energy offset between basins. Therefore, the energy offset (surface energy in this example) becomes more accurate as the temperature of the training set increases.

Following the above example, the intermediate regions between basins should be sampled to avoid ad hoc energy mapping. Alternatively, adding structures that have a different number of atoms in each basin can lift the ad hoc energy mapping. For instance, if the slab model is trained with the bulk crystal structures, or slab structures with different thickness, ad hoc energy mapping would have disappeared. However, such deliberate choice of the training set would be difficult when the system becomes more complicated.

For a systematic analysis, we develop a metric that measures the connectivity of training points in the symmetry function space. We carry out a single-linkage clustering of training points. The single-linkage clustering is a kind of hierarchical clustering method used in the statistical analysis. At the start, every training points are a cluster of size one. At each step, two clusters with the shortest distance merge into one. The distance between two clusters is set to the minimum Euclidean distance between two points from each cluster. We define r_g as the distance between the last two large clusters, whose size is larger than a threshold value. The threshold is set to half of the number of structures in the training set, but the result is largely insensitive to the threshold. As r_g gives the maximum distance among the groups of clusters, r_g^{-1} can be regarded as the connectivity between clusters.

Figure 3.4(a) shows r_g^{-1} against the temperature of training sets for the slab model.

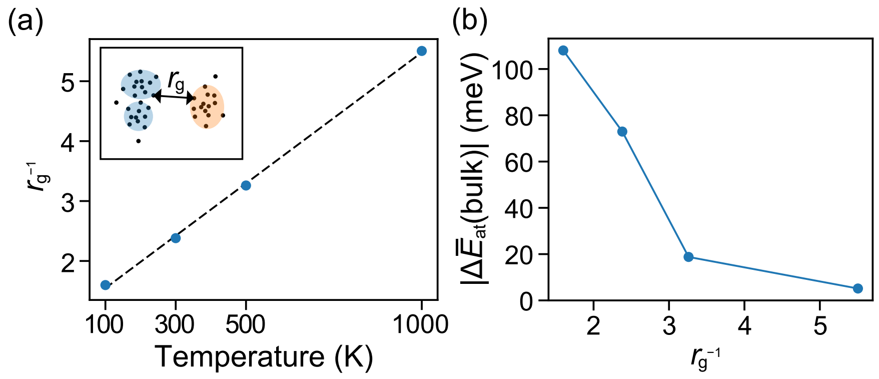


Fig. 3.4: (a) r_{g}^{-1} against the temperature of training set and (b) absolute atomic energy error of the bulk versus r_{g}^{-1} for the silicon slab model.

r_{g}^{-1} increases almost linearly with the temperature, supporting that the training set is more connected at a higher temperature. Figure 3.4(b) shows the relationship between $\Delta\bar{E}_{\mathrm{at}}(\mathrm{bulk})$ and r_{g}^{-1}. It is seen that the atomic energy mapping error decreases and converges to a sufficiently low value as r_{g}^{-1} increases. While the numbers would vary case by case, r_{g} can serve as a tool to examine the connectivity of the training set quantitatively. Thus, r_{g} can be used as a guide when constructing the training set.

3.3.4 Si nanocluster example

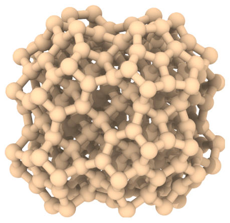
Albeit simple, the two examples in the previous sections demonstrate the ad hoc energy mapping that originates from the limitations in the training set. In practice, a training set usually includes various structures such as crystalline bulk, slab, and distorted crystals. Therefore, the ad hoc energy mapping demonstrated in the above examples can be avoided. Nevertheless, the error canceling between energy offsets of distinct configurations exists in almost any structures, and it might be unnoticed if the training process is monitored by only RMSE of total energy or atomic forces.

We demonstrate such a case in this example. The training set consists of MD snapshots of a 239-atom silicon nanocluster at 1000–1700 K. The structure of silicon cluster relaxed at 0 K is shown in Fig. 3.5(a). The training and validation set consist of 832 and 208 snapshots, respectively. The analysis of r_{g} confirms that the training points are well connected.

Figure 3.5(b) shows RMSE for the total energy and force with respect to the training epoch. It also shows $\Delta\bar{E}_{\mathrm{at}}(\mathrm{bulk})$ and $\Delta\bar{E}_{\mathrm{at}}(\mathrm{surface})$ as defined in Section 3.3.3. The analysis that is similar to the one in Section 3.3.2 indicates that the $\mathbf{G}$ points in the slab model are close enough to the training set such that they can be learned. Therefore, NNP is expected to predict the bulk and surface energies well. In Fig. 3.5(b), it is seen that RMSE for total energy and atomic forces converges rapidly and remains almost constant after about 100 epochs, while $\Delta\bar{E}_{\mathrm{at}}(\mathrm{bulk})$ and $\Delta\bar{E}_{\mathrm{at}}(\mathrm{surface})$ converge much slower. This shows the risk of terminating the training process by the convergence of RMSE for total energy and atomic forces, as people usually do. Therefore, it is recommended to monitor invariant quantities during the training process to assess the quality of atomic energy mapping. Obviously, if we include the crystalline and slab structures in the training set, $\Delta\bar{E}_{\mathrm{at}}(\mathrm{bulk})$ and $\Delta\bar{E}_{\mathrm{at}}(\mathrm{surface})$ would converge much faster, but that does not guarantee the overall quality of atomic energy mapping.

After sufficient training epochs, the surface energies for (100)-(2×2), (110)-(2×1), and (111)-(2×1) are in good agreement with DFT within 8 % (structures are relaxed

(a)



(b)

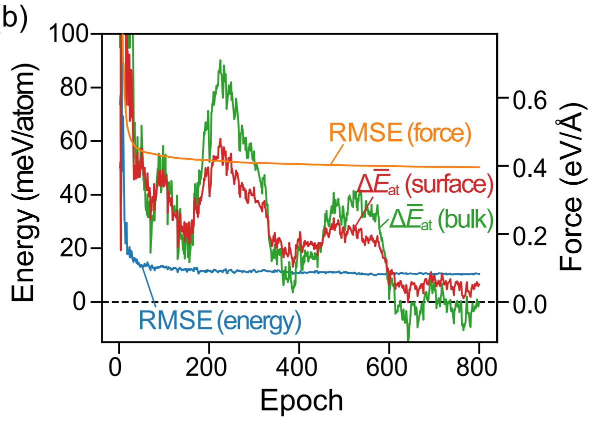


Fig. 3.5: (a) Structure of Si_{239} nanocluster relaxed at 0K. (b) Convergence of RMSE for total energy and atomic forces, and atomic energy mapping errors for surface and bulk regions of Si(100)-(2×2) slab with respect to the training epoch.

with NNP and DFT independently), whereas the errors are within 20 % at 200 epoch. This implies that NNPs with proper atomic energy mapping are more transferable than those with ad hoc energy mapping.

3.4 Implication for multi-component system

In multi-component systems, the sum of E_{at} over the high-symmetry motif is uniquely defined in DFT such that it can be used as reference values to test the atomic energy mapping of NNP. For instance, $E_{\text{at}}(\text{Mg}) + E_{\text{at}}(\text{O})$ in fcc MgO is uniquely defined. However, the relative offsets among different chemical species can be arbitrary values. Nevertheless, the (ensemble average) atomic energy relative to its unary phase has physical meaning (related to the energy of mixing), and it should be in a reasonable range.

If the training set consists of only structures with single stoichiometry, the energy offset becomes completely arbitrary. For example, $E_{\text{at}}(\text{Ge}) + \Delta$ and $E_{\text{at}}(\text{Te}) - \Delta$ in GeTe produce exactly the same total energies and atomic forces even for unreasonable values of Δ (where E_{at} is defined relative to the unary phase). This corresponds to the ad hoc energy mapping along with the composition in multi-component systems, which can undermine the stability of simulations. For instance, we found that MD simulations of liquid GeTe are often plagued by unphysical phase separation. The phase separation occurs when the training set consists of only 1:1 composition, or several compositions near 1:1 but without unary phases. The problem can be overcome by including structures with various compositions from 0:1 to 1:0. The atomic energy is well defined at the end compositions, and the intermediate compositions connects the atomic energies among different compositions.

This is analogous to the ad hoc energy mapping in Section 3.3.2, but the ad hoc energy mapping is along with the local composition rather than the local density. There is local fluctuation in the composition, but they cancel out in the training set with a single composition. Note that this does not mean that atomic energies are uniquely defined at mixed compositions (only the average atomic energy is uniquely defined).

The effect of continuous sampling over composition space is demonstrated in Fig. 3.6 for GeTe system. When the training set consists of only compositions near 1:1, ad hoc energy mapping occurs, and unphysical phase separation occurs [Fig. 3.6(b)]. However, when the whole composition range is sampled with the trajectory of interface reaction, ad hoc energy mapping disappears, and the liquid phase is well described without the phase separation issue.

Due to the additional degree of freedom in the composition, ad hoc energy mapping is more prevalent in multi-component systems, which can lead to stability issues. Therefore, more care needs to be taken for such systems.

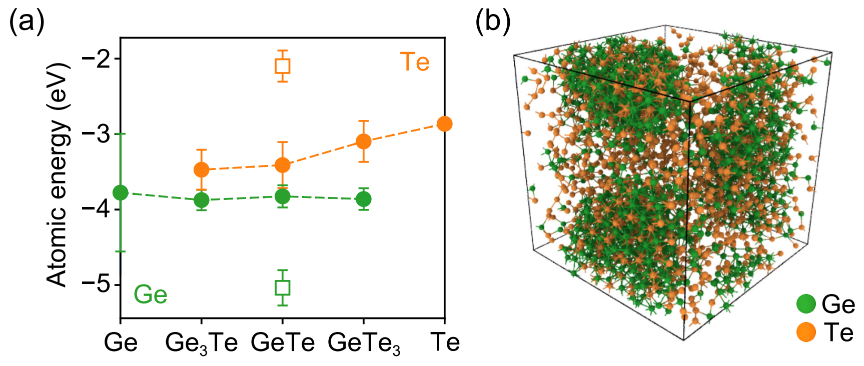


Fig. 3.6: (a) Average atomic energies of Ge and Te in the liquids of various compositions. Filled circles are results when the training set includes Ge, Ge_3Te, GeTe, $GeTe_3$, and Te. The empty squares are results when the training set includes only compositions near 1:1. Error bars indicate one standard deviation in atomic energies for MD snapshots. (b) The snapshot with unphysical phase separation, which occurs when NNP is trained over only compositions near 1:1.

3.5 Summary

To summarize, we investigated the atomic energy mapping of neural network potentials. We first showed that the transferable atomic energy can be defined within DFT. The existence of transferable atomic energy in DFT implies that the objective of NNP is to learn the underlying atomic energy function defined at the DFT level when only total energies are informed. The transferability of NNP lies in the accuracy of atomic energy mapping. The examples consistently support that NNP is capable of learning underlying atomic energy function from total energies. On the other hand, it was also observed that NNP is vulnerable to ad hoc energy mapping, where NNP predicts the total energy accurately for the given training set, but gives markedly wrong atomic energy function. The examples show simple cases where the constraints on the molecular dynamics simulations, such as fixed volume, fixed composition, or a fixed number of atoms in each configuration, leads to ad hoc energy mapping. The atomic energy mapping can be improved by choosing the training set carefully (removing constraints or improving connectivity) and monitoring the atomic energy mapping with some invariant quantities during the training process. The implication for multi-component systems is also discussed. Although we have focused our discussion on NNP, the conclusion is generally applicable to other machine learning potentials that are based on the locality (eqn 3.1).

Chapter 4

Metadynamics sampling

4.1 Introduction

One of the most frequent obstacles to the development of neural network potential is that it is prone to failures during the molecular dynamics simulation. The *failure* indicate the situation where atoms visit untrained (undersampled) area in the symmetry function vector space, and then the system evolves to the states with unphysically high energy. The failure happens because NNP cannot extrapolate well, given no physical background.

For the conventional applications, one needs to construct a training set that should contain all possible local environments that can appear during the simulations. However, knowing what structure would emerge during the simulation *a priori* is difficult even with intuition, experience, and the knowledge about the target system. The usual procedure of developing NNP has been a tedious trial and error process. First, one constructs the initial training set and train neural network potential. Second, one performs simulations and observes failures. Third, one reconstructs the training set so that it includes the new local environment that emerged during the last simulation. This process can be repeated many times. This iterative procedure takes a huge amount of time and effort. Furthermore, those failures might be rare events or slow process so it might take a long time before one observe failures in the simulation. More importantly, it can silently fail without any noticeable indications. Therefore, much study has focused on the uncertainty estimation and the detection of failures during the simulation.[42, 76–79]

Gaussian approximation potential tries to detect this problem by predicting the confidence interval, which is available as a part of the Gaussian process regression model. NNP has no such built-in confidence interval, so other workarounds were suggested. The ensemble of NNP is one way to measure the prediction uncertainty of NNP.[42, 76, 77] In the method of ensemble, several independent NNPs (preferably with different network structures) are trained. The independent NNPs would give similar results for the training set, but their prediction would diverge when they are giving a prediction for the untrained region. Then, the standard deviation of the prediction value of NNPs is proportional to the prediction uncertainty. However, the method of ensemble needs several NNPs to be trained, requiring a much higher computational cost.

Another popular approach to the problem of constructing a training set is to never stop the training. Active learning, or so-called *on-the-fly* method, trains models iteratively while updating the training set during the simulation. Active learning is widely adopted throughout the studies.[76, 78–80] It utilizes the method of uncertainty estimation described above to check if *ab initio* calculation is needed or not. If machine learning potential is accurate enough, the simulation is proceeded with machine learning potential. Otherwise, *ab initio* calculation is performed, and the results are added to the training set. As the simulation goes on, a machine learning potential is progressively improved, and less and less *ab initio* calculations would be required.

However, active learning has some limitations. First, the simulation is limited to small size because one might have to perform *ab initio* calculation. This severely limits the full potential of machine learning potentials. Also, the employment is much difficult, because then *ab initio* code and MD code have to be coupled, and system running

MD also need the capability to run *ab initio* calculation, where the suitable system architecture for MD and *ab initio* calculation may differ. Moreover, active learning is not a solution to the problems of sampling methods, rather a complementary method to them; Active learning must be performed with other sampling methods and suffers the same problems.

A systematic and efficient sampling method that can generate a robust training set can greatly cut the development time and make machine learning potential easily accessible. In this vein, a few sampling methods were suggested,[28, 81, 82] but still, the problem of sampling remains the main challenge. In the following section, we briefly review some of the sampling methods proposed and suggest a novel metadynamics sampling scheme.

4.2 Sampling methods

Molecular dynamics

Molecular dynamics (MD) is the most common method to generate a training set. Multiple snapshots of a certain interval are collected and used as a training set. If one generates a training set from MD at the target application temperature, resulting in machine learning potential becoming too fragile as the high energy configurations (which occurs due to thermal fluctuations) are rarely sampled. Therefore, it is common to perform MD at high temperatures to capture high-energy configurations and increase stability. The advantage of using MD sampling is that it is simple and readily available in various codes.

However, MD sampling has a few disadvantages. First, the sampling is biased toward the low-energy region, so high energy states are undersampled. This also means that sampling configurations that are separated by high energy barriers or rare events are very challenging or even impractical. Second, the high-temperature strategy may not be feasible for soft materials such as organic molecules because they can break apart at high temperatures. Lastly, the training set generated by MD sampling is highly redundant. In many cases, most snapshots have the same distribution of local environments or the same characteristics, which gives little information. Despite these disadvantages, it is still a popular method for its simplicity. In many cases, MD simulation itself is also our target application. Therefore, MD is often combined with on-the-fly method.

Typically, the prediction performance of a machine learning model increases and converges at some point as the number of data increases. However, increasing the number of data by reducing sampling interval would not increase the prediction performance much at some point, because the sequential snapshots are very similar and not much new information is contained in the new snapshots. Due to the redundancy of molecular dynamics sampling, the prediction performance converges with a smaller number of snapshots than other sampling methods.

Normal mode sampling

Normal mode sampling (NMS) was proposed by J. S. Smith et al. in the paper developing neural network potential for organic molecules.[28] NMS was suggested to obtain physically relevant structures in an accelerated way without long redundant molecular dynamics simulations. First, all structures are relaxed. Next, N_f normal mode coordinates and corresponding force constants are calculated for relaxed structures. Then, N_f uniformly-distributed pseudo-random numbers c_i are generated such that $\sum_i^{N_f} c_i$ is in the interval $[0, 1]$. Then, we set the harmonic potential due to displacement for ith normal mode, R_i, is equal to the average energy at temperature T scaled by c_i:

$$\frac{1}{2}K_i R_i^2 = \frac{3}{2}c_i N_{\text{atom}} kT\,, \tag{4.1}$$

where K_i is the force constant corresponding to ith normal mode. Solving for R_i gives:

$$R_i = \pm\sqrt{\frac{3c_i N_{\text{atom}} k_b T}{K_i}}\,. \tag{4.2}$$

The sign of R_i is determined randomly. Now, we can generate distorted high-energy structures by the superposition of R_i. NMS generates samples uniformly along the energy scale, increasing the stability of machine-learning potential. It is suitable for systems where the local vibration within a well-defined structure dominates the dynamics such as crystals and some molecules.

There are downsides to the method. First, NMS requires expensive normal mode calculations, which can be prohibitive with large systems. Second, it might not be suitable for systems that are not well represented by a few minima (e.g., liquid and amorphous). Also, as pointed out in Ref. [82], it is based on the harmonic approximation, so it might not be suitable for large distortions that show strong anharmonicity. Another important weak point is that NMS cannot be used to explore and sample other metastable configurations or barriers.

Random structure search

Random structure search (RSS) was originally suggested as a method to perform structure searching (finding stable structures for a system with unknown structures).[83] The basic idea of RSS is simple. We first generate random (but sensible) structures. Then, each random structure is minimized to find the most stable structure in each basin. The *sensible* structures are generated by imposing some constraints such as symmetry, bond lengths, chemical senses to bias searches toward low energy minima. Ref. [83] used *ab initio* calculation to minimize structures, so it was named as *ab initio* random structure search (AIRSS).

The idea of the random structure search was later adopted in the application of developing machine learning potential.[81, 84] In their approach named GAP driven random structure search (GAP-RSS), they first generate a machine learning potential (GAP) from random structures. Then, a large set of random structures are relaxed with the machine learning potential. The relevant minima are selected considering energy and diversity. Again, the representative structures are selected from the minimization trajectory leading to those pre-selected structures. The next generation of machine learning potential is generated, and the whole process is repeated until a certain number of structures are collected.

The advantages of GAP-RSS are as follows: first, it can explore a wide range of local minima. Second, by imposing energy criteria in the structure selection, energetically relevant structures are collected. Third, it is efficient as we only perform a minimal amount of *ab initio* calculations as we perform minimization with the machine learning potential.

The downsides are as follows: first, it requires an iterative refinement of machine learning potential like active learning methods, which can be costly, especially for neural-network-based potentials. Second, the minimization trajectory is generated from the potential energy surface of machine learning potential, which could be less relevant to the potential energy surface of DFT. Lastly, although RSS is great for searching bulk crystalline structures, it might not be optimal for learning dynamics of atoms and sampling disordered or non-periodic systems.

Metadynamics sampling

Metadynamics[55] is a powerful technique that enhances the sampling of molecular dynamics simulations. A detailed description of metadynamics is given in Section 2.3. In this section, we describe the applications of metadynamics to the generation of data set for machine learning potentials. We first introduce the previous related study and then propose our novel metadynamics sampling scheme.

The first application of metadynamics for machine learning potential was suggested by J. E. Herr et al. [82] They assessed the effectiveness of MD, NMS, and metadynamics sampling for a nicotine molecule and a ten-water-molecule cluster. They used high-dimensional neural network potential with the ANI-1 modified symmetry functions[28], and a distance matrix ($D_{ij} = 1/R_{ij}$) is used as a collective variable. Thus, the bias potential is defined as:

$$U_b = \sum_{i=1}^{N_{\text{sample}}} h \exp\left(-\frac{|\mathbf{D} - \mathbf{D}_i|^2}{2\sigma^2}\right) , \tag{4.3}$$

where N_{sample} is the number of all visited structures. They found that MD sampling is unacceptably poor, while NMS and metadynamics sampling gives comparable results.

However, the method using a distance matrix as a collective variable suffers from problems. First, it is difficult to apply the method to the periodic or large system because a distance matrix is not well defined in the periodic system and the size scales to N_{atom}^2. Second, all distances are equally treated while they are not equally important. This can cause redundant sampling. Lastly, the sampling can be inefficient as CV components can be degenerate. For example, If two water molecules are far away, the distances between the two hydrogens are degenerate. Even if the H-H distance is well sampled in one water molecule, it is not reflected on the H-H distance in the other molecule. Therefore, one is double sampling the equivalent collective variables.

In this dissertation, we propose a novel metadynamics method, which directly use a descriptor vector, which is fed into the machine learning model, as a collective variable. Here, we abbreviate the descriptor vector as $\mathbf{G}$. For instance, if a descriptor vector is 30-D, CV is 30-D, too. Unlike conventional metadynamics, where CV describes the state of the whole system, CV describes the local environment of each atom in the suggested method. The bias potential is constructed in the descriptor space, and all atoms of the same chemical species share the bias potential. The bias potential added to the system is a sum of atomic bias potential in resemblance to eqn 2.15:

$$U_{b,\text{tot}} = \sum_{i=1}^{N_\text{atom}} U_b\left(\mathbf{G}_i\right) \ . \tag{4.4}$$

By using a local descriptor as a collective variable and then defining total bias potential as a sum of local contributions, we can explore the local environment space for each atom rather than the configuration space of the whole system.

Since we use a local descriptor, the application to periodic or large system is trivial (the dimension of CV is constant regardless of the system size). Also, by sharing bias potential between equivalent atoms (of the same chemical species), the problem of degeneracy vanishes. As pointed out in Chapter 3, machine learning potential learns transferable atomic energy as a function of the local environment (local descriptor). Therefore, enhancing sampling directly on the descriptor space would improve the training without the problem of redundancy.

The similar form of *sharing* bias potential was suggested recently.[61] They pointed out the problem of inefficient sampling of degenerate CVs. Therefore, they grouped multiple identical CVs into a family, inside which the same bias potential is shared. The method is based on the parallel bias metadynamics, so it is named parallel bias metadynamics with partitioned families (PBMetaDPF). Our method can be regarded as a kind of partitioned family method (but not parallel bias metadynamics). The whole system can be considered to be described by $N_G N_\text{atom}$ collective variables, but all atoms of the same chemical species are indistinguishable, so their CVs are degenerate.

Therefore, all collective variables that correspond to the same symmetry function are grouped into a family, where they share the same bias potential.

Using geometry-adapted Gaussians [62] (see Section 2.3.1 for details), the bias potential of an atom is defined as:

$$U_b\left(\mathbf{G}\right) = \sum_{i=1}^{N_{\text{sample}}} h \exp\left[-\frac{1}{2}\sum_{j=1}^{N_G}\sum_{k=1}^{N_G}\left(G_j - G_j^{(i)}\right)\sigma_{jk}^{-2,(i)}\left(G_k - G_k^{(i)}\right)\right], \tag{4.5}$$

where N_{sample} is the number of all sampled $\mathbf{G}$'s and N_G is the dimension of the descriptor vector, and $G_j^{(i)}$ is jth component of the descriptor vector of ith sample. σ_{jk}^2 is the covariance between jth and kth descriptor components. It depends on the value of $\mathbf{G}$ and computed as:

$$\sigma_{jk}^2(\mathbf{G}) = \sigma_G^2 \sum_{i=1}^{N_{\text{atom}}} \sum_{\alpha=x,y,z} \frac{\partial G_j}{\partial R_{i,\alpha}} \frac{\partial G_k}{\partial R_{i,\alpha}}, \tag{4.6}$$

where α indicates three axes of Cartesian coordinates, and σ_G is a hyperparameter that controls the width of Gaussian potential. The biasing force acting on each atom is computed as:

$$F_{i,\alpha} = -\frac{\partial U_{b,\text{tot}}}{\partial R_{i,\alpha}} = -\sum_{j=1}^{N_{\text{atom}}} \sum_{s=1}^{N_G} \frac{\partial U_{b,i}}{\partial G_{j,s}} \frac{\partial G_{j,s}}{\partial R_{i,\alpha}}. \tag{4.7}$$

The geometry-adapted Gaussians are adopted because the components of a descriptor vector are often highly correlated. Especially, atom-centered symmetry functions are highly correlated to each other. Use of isotropic Gaussians (eqn 2.50) results in inefficient sampling in such a case. If one uses a small σ, then it takes too much time to fill and explore the energy surface. Whereas, directions with small variance are effectively ignored if one uses a large σ. Moreover, the *volume* swept by the same degree of perturbation to atomic coordinates is position-dependent. This leads to biased sampling toward the region where the *volume* is larger. For instance, a radial symmetry function (assuming $R_s = 0$) has a larger value at higher atom density. This means that, given random displacements to nearby atoms, the change in the symmetry function value is also larger at higher atom density. Therefore, the sampling is biased toward high-density configurations, because it takes more time to fill the region with higher density. For such reasons, it is recommended to use adaptive Gaussians for correlated descriptors, despite of increased computational cost and complexity.

We also suggest a variation named *partial bias* metadynamics. In the partial bias metadynamics, only a set of a few selected atoms contribute to the total bias potential:

$$U_{b,\text{tot}} = \sum_{i=1}^{N<N_\text{atom}} U_b\left(\mathbf{G}_i\right) \ . \tag{4.8}$$

The partial bias metadynamics is suggested to enhance the sampling of defective structures embedded in the crystalline bulk. Since every atom contributes to the bias potential in the original scheme, the biasing force drives all atoms out of crystalline structure simultaneously. Therefore, it can be difficult to sample defects where other surroundings are still in the crystalline phase. The partial bias metadynamics tries to address such a problem by applying atomic bias potential only to a few selected atoms (which are spatially separated) so that the selected atoms search new configurations while the majority remains in the crystalline state. The downside is that the sampling is more biased toward the crystalline phase like molecular dynamics sampling, which could be handled with GDF weighting[13] (see Section 2.2.4).

One might be concerned about the high dimensionality of a descriptor vector. Since the time to sample N-dimensional space would increase exponentially (for instance, sampling a grid of five points along ten dimensions already requires $5^{10} \approx 10^7$ points), it is often advised to keep a small set of CV. However, only a tiny fraction of the entire volume of the descriptor space is mathematically possible (not all $\mathbf{G}$ have the corresponding local environment), and the small fraction of that tiny volume is energetically accessible. Furthermore, the regularization on a covariance matrix σ_{jk}^2 effectively limits the dimension one explores; the exploration is mainly focused on the first few principal components, and the lesser components are effectively ignored. Therefore, the sampling of $\mathbf{G}$-space is feasible despite its high dimensionality.

There are three hyperparameters in the method: h, σ_G, and Δt. h and σ_G control the height and the width of the Gaussian potential, respectively. Combined, they determine the strength of biasing force. σ_G determines a similarity measure between structures. When one uses too small σ_G, it would take too much time to fill the energy surface. In contrast, too large σ_G value would reduce the overall effectiveness of metadynamics. If an empirical potential is available for the target system, σ_G can be determined with preliminary tests using the empirical potential. The value of h is determined so that the magnitude of the biasing force is in a reasonable range. If it is too small, the biasing force has no effect. If it is too large, the biasing force is so large that the simulation becomes unstable. Δt is the interval of adding samples. If Δt is too small, not only the computational cost is prohibitive, but also the dynamics are severely affected. A sensible choice of Δt is the interval of sampling data set from the trajectory. The choice of hyperparameters are mostly transferable to other systems. One can adjust h and σ_G according to the cohesive energy or atomic density of a new target system.

There is a resemblance between eqn 2.15 and eqn 4.4, and between eqn 2.16 and eqn 4.7. The only difference is whether evaluate atomic energy $E_{\text{at},i}$ or atomic bias potential $U_{b,i}$. Therefore, the metadynamics method can be implemented in a straightforward manner by swapping neural network to a bias evaluation part. We implemented a pair style that computes the bias potential to LAMMPS molecular dynamics package[85]. One can interface LAMMPS with other *ab initio* simulation packages such as VASP to perform *ab initio* metadynamics simulation. The interfacing can be done through either calling VASP and LAMMPS in ASE[86] or LAMMPS client calling VASP server (LAMMPS support server-client model).

For the remaining part of this chapter, we demonstrate the applications of new metadynamics method with the neural network potentials, using an atom-centered symmetry function vector as a collective variable.

4.3 Computational details

Generation of training set

Metadynamics simulations are performed with LAMMPS[85], which is coupled with an *ab initio* calculation code (VASP) using a server/client model. The parameters for metadynamics simulations are chosen with preliminary tests using classical potentials. (We note that metadynamics simulation is rather insensitive to the choice of symmetry function parameters) Multiple metadynamics simulations are carried out with different settings, such as starting structure, metadynamics parameters, and pressure, to cover a much wider range of configurations efficiently and obtain a data set of sufficient size. Initial structures are taken from a few well-known stable crystalline structures and stable slab structures. The training set contains a few 20–40 ps metadynamics trajectories with the timestep of 2 fs. For silicon, we added thirty distorted diamond structures and an isolated atom. We include distorted crystals to give a better description of elastic properties because elastic constants show high error when trained only with dynamics trajectories. Detailed information on the metadynamics data set is given by Table 4.1 for silicon and Table 4.2 for aluminum. The data set is split randomly into a training set and a validation set with 95:5 ratio. Using a larger σ_G, one can sample a wide range of configurations more quickly, whereas a small σ_G is used to sample more of relevant configurations. The high-pressure simulation is included for silicon to cover high-pressure phases. Partial bias simulations and the simulations starting from defective crystals are included specifically to enhance to sampling of defective structures.

All density functional theory calculations were performed with plane-wave based Vienna *ab initio* simulation package (VASP)[87–89] using projector-augmented-wave pseudopotentials[90]. Generalized gradient approximation with Perdew-Burke-Ernzerhof (PBE) functional[91] is used for the description of exchange-correlation energy of electrons. We perform one-shot calculations on the sampled trajectories at higher calculation settings. The calculation settings such as cutoff energy for plane-wave basis

Table 4.1: Metadynamics data set used to generate general-purpose neural network potential for silicon. Each item is a metadynamics trajectory of $20–40\,\mathrm{ps}$ with the given setting. Snapshots are collected in $40–80\,\mathrm{fs}$ interval. The entire data set is randomly split into the training set and the validation set with 95:5 ratio. “Diamond (defects)” indicates the diamond crystal structure with one vacancy and one hexagonal interstitial atom.

Starting structure	Note	h (eV)	σ_G (Å)	Δt (fs)	T (K)	#structure	#atom
Diamond		0.0006	2.5	20	300	489	31 296
Diamond		0.0002	2.5	20	300–2500	249	15 936
Diamond (100)		0.0002	1.5	20	300	127	8128
Diamond (100)		0.0003	2.5	20	300	98	6272
Diamond (defects)		0.0002	2.5	20	300	80	5120
Diamond (defects)		0.0006	1.5	20	300	121	7744
Diamond	10-20 GPa	0.0001	0.6	20	300	1001	64 064
Diamond	Partial bias	0.04	4.0	20	300–1000	315	20 160
Diamond	Partial bias	0.02	1.5	20	300	407	26 048
Diamond		0.0006	0.4	20	300	737	47 168
Diamond		0.0002	1.5	20	300	1940	124 160
Diamond (100)		0.0006	0.4	20	300	620	39 680
Diamond (100)		0.0002	1.5	20	300	372	23 808
Diamond (111)		0.000 45	0.4	20	300	953	45 744
Total						7509	465 328

Table 4.2: Metadynamics data set used to generate general-purpose neural network potential for aluminum. Each item is a metadynamics trajectory of 20–40 ps with the given setting. Snapshots are collected in 40–80 fs interval. The entire data set is randomly split into the training set and the validation set with 95:5 ratio.

Starting structure	Note	h (eV)	σ_G (Å)	Δt (fs)	T (K)	#structure	#atom
FCC		0.000 25	0.5	20	300	588	63 504
BCC		0.0005	0.5	20	300	921	49 734
HCP		0.000 42	0.5	20	300	844	54 016
FCC (100)		0.000 25	0.5	20	300	542	58 536
FCC (111)		0.000 25	0.5	20	300	568	54 528
FCC (111)		0.000 25	1.5	20	300	1242	119 232
Total						4705	399 550

and the number of k-point are determined by convergence tests. The Monkhorst-Pack grid is used to sample k-points with the spacing between k-points in the reciprocal space specified. The settings were set such that the total energy RMSE converges within $10\,\mathrm{meV/atom}$, and the force RMSE converges within $0.03\,\mathrm{eV/Å}$ for randomly sampled snapshots. For silicon, we used a cutoff energy of $350\,\mathrm{eV}$ and k-point spacing of $0.157\,\mathrm{Å}^{-1}$. For aluminum, the cutoff energy of $250\,\mathrm{eV}$ and k-point spacing of $0.2\,\mathrm{Å}^{-1}$ were used for one-shot calculations. Snapshots with low convergence (too many electronic steps) or too large forces ($> 6\,\mathrm{eV/Å}$) are discarded to improve the quality of the final data set.

Training of neural network potential

Symmetry function parameters are selected using CUR method[9, 39] (see Section 2.2.2) from a large initial pool of 233 parameters that are systematically constructed. In the process of CUR, each symmetry function is weighted by the estimated evaluation cost, which depends on the cutoff radius and whether it is a radial or an angular function so that the most cost-effective set of parameters is selected. CUR is performed on the diverse data set generated from a silicon metadynamics trajectory with Stillinger-Weber potential. The CUR selection is terminated when the error drops below $0.2\,\%$, resulting 47 parameters. The same parameter set is used for both silicon and aluminum as we note that a good parameter set for one system is mostly transferable to other systems. The final parameter set used in the following applications is listed in Table 4.3.

The structure of the atomic neural network is set to two hidden layers with 120 (90) hidden nodes each, i.e., 47-120-120-1 (47-90-90-1) network, for silicon (aluminum). The number of hidden nodes per layer is optimized to reduce the error without overfitting. The training is performed with momentum-based Adam optimizer[50] with minibatch (batch size of 20 snapshots), which gives good performance under reasonable computational cost. An exponentially decaying learning rate is used to improve the convergence. We found it is efficient to employ a multi-step procedure, where a large force coefficient is used in the first step, and a small or zero force coefficient (with a lower learning rate) is used in the second step. In the first step, force error is gradually reduced, but energy error fluctuates widely. In the second step, energy error converges quickly while the force error increases by a small amount. The force and stress coefficient μ and ν (defined in eqn 2.46) used in the first step is $0.1\,\text{Å}^2$ and 0.0005–$0.001\,\mathrm{eV}^2/\mathrm{kbar}^2$.

Five percent of the entire data set is randomly collected as a validation set. The overfitting, where the error for the validation set increases, is monitored during the training process, and the early-stopping method is used in the case of overfitting.

Table 4.3: Symmetry function parameters

Type	R_c (Å)	η (Å^{-2})	ζ	λ
radial	6.5	0.000		
radial	6.5	0.059		
radial	6.5	0.131		
radial	5.0	0.000		
radial	5.0	0.131		
radial	5.0	0.253		
radial	3.5	0.000		
radial	3.5	0.090		
radial	3.5	0.253		
radial	3.5	0.464		
radial	3.5	0.623		
ang.n.	6.5	0.000	1	-1
ang.n.	6.5	0.000	4	-1
ang.n.	6.5	0.035	4	-1
ang.n.	6.5	0.000	8	-1
ang.n.	6.5	0.035	1	1
ang.n.	6.5	0.000	2	1
ang.n.	6.5	0.000	8	1
ang.n.	6.5	0.000	16	1
ang.n.	6.5	0.035	16	1
ang.n.	5.0	0.035	1	-1
ang.n.	5.0	0.000	2	-1
ang.n.	5.0	0.090	2	-1
ang.n.	5.0	0.000	4	-1
ang.n.	5.0	0.000	8	-1
ang.n.	5.0	0.000	4	1
ang.n.	5.0	0.090	4	1
ang.n.	5.0	0.000	16	1
ang.n.	5.0	0.184	16	1
ang.w.	5.0	0.000	1	-1
ang.w.	5.0	0.090	1	-1
ang.w.	5.0	0.000	2	1
ang.w.	5.0	0.184	4	1
ang.w.	5.0	0.000	8	1
ang.w.	5.0	0.090	8	1
ang.w.	5.0	0.000	16	1
ang.w.	5.0	0.090	16	1
ang.w.	3.5	0.344	1	-1
ang.w.	3.5	0.000	2	-1
ang.w.	3.5	0.184	2	-1
ang.w.	3.5	0.000	1	1
ang.w.	3.5	0.184	2	1
ang.w.	3.5	0.000	4	1
ang.w.	3.5	0.344	4	1
ang.w.	3.5	0.000	8	1
ang.w.	3.5	0.184	8	1
ang.w.	3.5	0.000	16	1

4.4 Applications

There can be many applications of metadynamics for the development of machine learning potentials. We suggest three main applications here.

First, metadynamics can be performed to catch unexpected failures of premade machine learning potentials. Since machine learning potentials cannot give reliable predictions to an unexplored region, it is possible that an unexplored high-energy region is predicted to have low energy, leading to a *hole* in the potential energy surface. This can lead to the emergence of high energy structural motifs, high energy phases, or even catastrophic failures.[13] Usual process to determine if premade machine learning potential is stable or problematic, one performs time-consuming molecular dynamics simulation for a few nanoseconds to check if any failure happens. However, since a failure can be a rare event, even if no failure is detected in the test run, the problem can be there in the production run. By doing metadynamics with premade machine learning potentials, we can catch those possible failures efficiently. As the metadynamics simulation proceeds, the barrier is accumulated, and the energy of the system increases until it finds a *hole* in the potential energy surface. Eventually, NNP would slip into an unexplored, low-energy region (failure). Additionally, the barrier height at the failure can give a measure of the stability of NNP.

The second one is to sample wide range of **G**-space systematically to develop general-purpose potential. In order to develop general-purpose potential, one must construct a huge data set which contains a vast range of local environment. This is a formidable task and requires deep knowledge about the target system. Unlike classical potentials, most of the machine learning potentials developed so far have been mostly like *develop once, use once*. Since machine learning potential can only describe the structures that are included in the training set, and it is difficult to construct a huge training set that covers a wide range of local configurations, machine learning potentials have been developed aiming very specific applications. Therefore, they cannot be used in other applications. There have been attempts to make general-purpose potentials, which can be distributed and used in most applications, with a carefully selected assortment of data set.[92, 93] We show that metadynamics sampling can sample a wide range of local environments automatically so that it can be used to develop general-purpose potentials.

The third application is to improve the stability of machine learning potential. It is closely related to the first application. The construction of a training set is a bottleneck for developing machine learning potential because when we train our model and perform simulations, it is very common that simulation fails with the appearance of unexpected configurations. If one can easily construct a robust training set, which results in stable potential, the development time can be cut significantly, and machine learning potential can be more easily accessible and widely adopted. The failure happens because conventional molecular dynamics sampling is biased toward low energy region, and high energy barriers and configurations are not sampled enough.[13] Metadynamics sampling can sample high energy structures by filling potential energy surface from the bottom, eliminating holes in the low energy region of the potential energy surface. J. E. Herr et al.[82] demonstrated the improvement of stability with his method of using a distance matrix.

In the following sections, we demonstrate three applications of metadynamics for developing machine learning potentials using neural network potentials.

4.4.1 Catching unexpected failures

The first application is a little bit different than other two applications. While the other two use metadynamics to generate a training set, the first application uses metadynamics to assess the quality of machine learning potentials. When we develop machine learning potential for a certain system, it is quite common that simulation fails as unexpected local environments emerge. It is one of the major bottlenecks hindering the development process. Prior to the application, we need to know whether the potential is appropriate for the simulation or not. If it gives configurations that have high energy in DFT due to inaccurate potential energy surface, the potential need refinement before it can be used in the simulation. Therefore, we first check the stability of machine learning potential we made by performing time-consuming molecular dynamics simulation for a few nanoseconds to find out if any failure occurs. However, a failure can be a rare event, which means that even if the potential passed the test, it could still give unreliable results during the actual application runs. Since metadynamics fills the potential energy surface from the low energy region, it can find any hole in the potential energy surface efficiently, driving the simulation to possible failures in a very short time.

During the development of neural network potential for GeTe liquid system (the details are not given here as it would be out of scope of the dissertation), we made several versions of potentials and found that some potential leads to unphysical short bonds or phase separation issues, while others do not. We performed metadynamics simulations with those potentials to catch any possible problems and compare the stability between potentials. The results are summarized in Fig. 4.1.

By performing metadynamics simulation with premade potentials, we found possible issues such as phase separation or unphysical short bond formation efficiently in a very short time compared to conventional molecular dynamics simulation. The blue line shows the potential that was plagued by the phase separation problem. As can be seen, it gives lower energy for phase-separated configurations, driving simu-

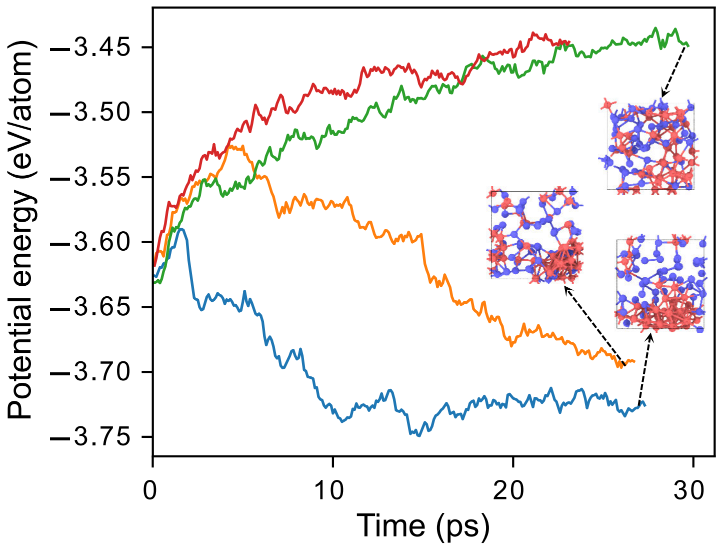


Fig. 4.1: Evolution of potential energy of premade neural network potentials (NNP) during metadynamics simulation. Different colors indicate different versions of NNP. Insets show the final configurations. The blue line shows phase separation and the orange line shows short bond formation ($\sim$1.6 Å) as well as phase separation.

lations to phase separation. The orange line in Fig. 4.1 shows the energy during the metadynamics simulation for the potential that showed no noticeable problem during molecular dynamics simulations. However, it was found to give lower energy for short bonds (bond length of about 1.6 Å) and phase separation. The barrier was higher than the blue one, so it would be a much rare failure. Thus, the failure was not detected in the test simulation. Nevertheless, it is likely for simulations to fail by the formation of short bonds at a higher temperature. Without metadynamics simulations, this kind of failure with high energy barriers (rare events) would be difficult to detect in advance.

We can also compare stability between potentials by looking at where those potentials fail; the one fails later at higher energy is more stable than the one fails earlier at lower energy. Green and red lines in Fig. 4.1 show no noticeable failures during the same period, indicating that they have higher stability than other two potentials.

For the case of short bond formation, the failure is obvious. However, sometimes the failure is more subtle and difficult to notice by just looking at it. We can check whether the simulation failed or not by calculating a few DFT energy along the trajectory and see if the gap between NNP energy and DFT energy suddenly increases or lose the correlation (see Section 4.4.3).

4.4.2 Developing a general-purpose neural network potential

Developing general-purpose machine learning potential is a formidable task. It involves the construction of a huge data set that contains a wide range of local configurations that can appear during the simulations, which requires human intuition and deep knowledge about the system and maybe several iterations of refinement. Recently, Botu et al.[92] developed a general-purpose NNP for aluminum, and Bartók et al.[93] developed a general-purpose GAP for silicon with a carefully selected data set, encompassing crystals, surfaces, defects, liquids, etc. They performed extensive tests on various properties. Still, it is difficult to guarantee if the data set contains all relevant configurations. Thus, they rely on uncertainty estimation to ensure that the test configuration is covered. When the target system becomes more complex systems, where its physics is not well known, or multiple elements are concerned, it becomes nearly impossible to manually construct a huge data set encompassing all relevant local configurations.

We suggested a metadynamics sampling method where the dynamics is altered such that it searches local environment space for each atom. The search is performed from low energy to higher energy configurations so that only energetically relevant configurations are sampled. Therefore, one can systematically sample a wide range of energetically relevant local configurations using metadynamics sampling without much human intuition or deep knowledge about the target system. The generated training set can be used to develop general-purpose potentials more easily.

In this section, we first demonstrate general-purpose neural network potential using Stillinger-Weber potential for silicon as a reference to show the possibility of constructing general-purpose potential with metadynamics sampling. Classical potential is used as a testbed because it has a clearly defined transfer range of atomic energy and simpler potential energy surface. Then, we demonstrate and validate general-purpose NNPs for silicon and aluminum, which are typical covalent and metallic systems, using DFT as a reference.

Stillinger-Weber potential

First, we develop general-purpose potential for silicon but using Stillinger-Weber potential [94] as a reference. Stillinger-Weber potential is a classical potential where the energy is defined as:

$$E = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j\neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk}) \,, \tag{4.9}$$

$$\phi_2(r_{ij}) = A_{ij}\epsilon_{ij} \left[B_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{q_{ij}} \right] \exp\left(\frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}} \right) \,, \tag{4.10}$$

$$\begin{aligned} \phi_3(r_{ij}, r_{ik}, \theta_{ijk}) = {} & \lambda_{ijk}\epsilon_{ijk} \left[\cos\theta_{ijk} - \cos\theta_{0,ijk} \right]^2 \\ & \times \exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}} \right) \exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik} - a_{ik}\sigma_{ik}} \right) \,, \end{aligned} \tag{4.11}$$

where r_{ij} is the distance between ith atom and jth atom, θ_{ijk} is the angle $\angle jik$, ϕ_2 is a two-body term, and ϕ_3 is a three-body term. The cutoff distance is given by $a \cdot \sigma$, which is 3.77 Å for the silicon potential we used.

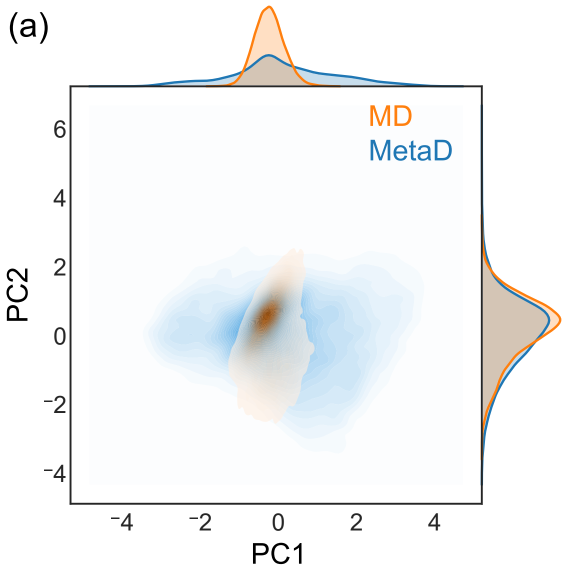
We first demonstrate a general-purpose potential using a classical potential as a reference for several reasons. First, the classical potential is cheap in terms of computational cost. We can quickly test and iterate through to check the potential of metadynamics sampling. It is appropriate for a preliminary test before we apply metadynamics to DFT calculations. Second, it does not suffer from transferability issues since the classical potential has a clearly defined transfer range. That is, atomic energy is precisely determined by the atomic arrangement within the specified cutoff radius. If the transfer range is not clearly defined, the error from using a finite cutoff radius can undermine the transferability of the neural network potential. Third, the potential energy surface of the classical potential is less complex than that of DFT. It would be easier for neural network potential to predict the potential energy surface of the classical potential. If the potential energy surface becomes too complex, the performance of the model can be limited by its representability.

The metadynamics simulation is performed with a few different settings, all starting from the diamond crystal structure. A metadynamics simulation with small σ_G samples the vicinity of crystalline structures, while a metadynamics simulation with large σ_G samples more various configurations such as amorphous, surfaces, and corner atoms of clusters. We also included a metadynamics simulation with high pressure so that high-pressure polymorphs could be sampled, and partial bias simulation to enhance sampling over defective structures.

To demonstrate that the metadynamics indeed samples a wide range of local configurations, we plot the sampling density of metadynamics sampling and high-temperature molecular dynamics sampling (1500–3000 K) in Fig. 4.2. The sampling density is evaluated over the two principal components of $\mathbf{G}$ (two combinations of principal axes are shown to give a more clear picture). It can be clearly seen that metadynamics sampling covered a much wider range in $\mathbf{G}$-space. To be specific, metadynamics covered dense configurations and the configurations that are similar to surfaces and corners of nanoclusters, which could not be sampled with high-temperature molecular dynamics.

The neural network potential trained with metadynamics training set is validated by comparing a number of static properties such as elastic properties, surface energies, and defect formation energies to the reference using a modified version of the testing code (silicon-testing-framework) from Ref. [93]. The results are listed in Table 4.4. Additionally, the equation of states for polymorphs, energy-strain curves, the energy along stacking fault, and the energy along the vacancy migration path are plotted in Fig. 4.3. The results are rather striking. Trained with only a few metadynamics simulation, the neural network potential gives almost perfect results (within $3\,\%$ error) for all properties that are tested, although those properties are not explicitly included in the training set.

One thing to note is that atomic energy is not uniquely defined in Stillinger-Weber potential. Although all properties were in great match, the individual atomic energy of neural network potential and Stillinger-Weber potential do not have a good correlation



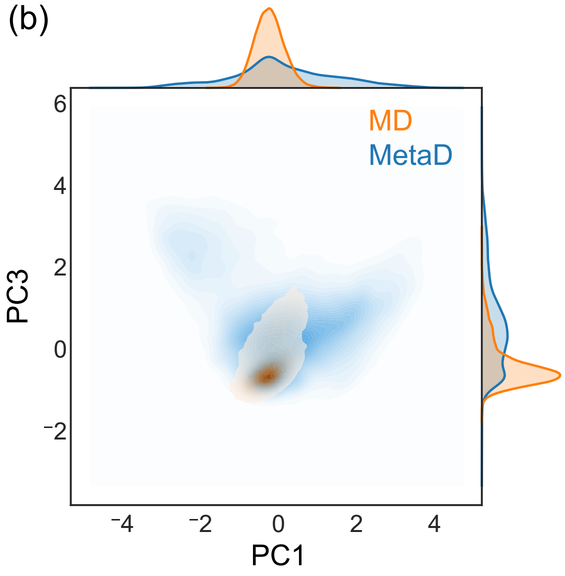


Fig. 4.2: Sampling density of high-temperature molecular dynamics (1500–3000 K; orange) and metadynamics sampling (blue) for Stillinger-Weber potential. The density is shown in (a) the first and second principal components and (b) the first and third principal components. Additionally, the density projected onto one axis is plotted over the axis. All simulations start from the diamond crystal structure.

Table 4.4: Comparison of various static properties between the reference Stillinger-Weber potential (SW) and the neural network potential (NNP) for silicon. Elastic constants are computed for diamond structure. Surface energies are calculated for (100) (2×2) reconstruction, (110) (1×1) reconstruction, and (111) (3×3) dimer-adatom-stacking-fault (DAS) reconstruction. The abbreviations vac., hex., tetra., db., and ffcd. stand for vacancy, hexagonal interstitial, tetragonal interstitial, dumbbell interstitial, and fourfold coordinated defect, respectively. GB indicates (112) $\Sigma 3$ grain boundary. $\gamma_{\text{us}}^{(s)}$ and $\gamma_{\text{us}}^{(g)}$ are unstable stacking-fault energies on shuffle plane and glide plane of diamond (111) planes.

		Elastic constant (GPa)				Surf. energy (J/m^2)			Defect formation energy (eV)					Planar def. (J/m^2)			Di-interstitial (eV)				
Model	a_0 (Å)	B	c_{11}	c_{12}	c_{44}	(111)	(110)	(100)	vac.	hex.	tetra.	db.	ffcd.	GB	$\gamma_{\text{us}}^{(s)}$	$\gamma_{\text{us}}^{(g)}$	XX3	EXT	W	XEX	XT
SW	5.43	101.8	151.7	76.9	53.4	2.04	1.67	1.44	2.83	6.76	5.19	4.62	3.02	1.24	0.87	3.09	6.70	6.15	5.74	6.27	5.38
NNP	5.43	104.1	156.9	77.8	57.8	2.04	1.67	1.44	2.86	6.65	5.23	4.62	3.02	1.24	0.87	3.08	6.74	6.24	5.80	6.09	5.44
%error	0	2	3	1	3	0	0	0	1	−2	1	0	1	0	0	0	1	1	1	−3	1

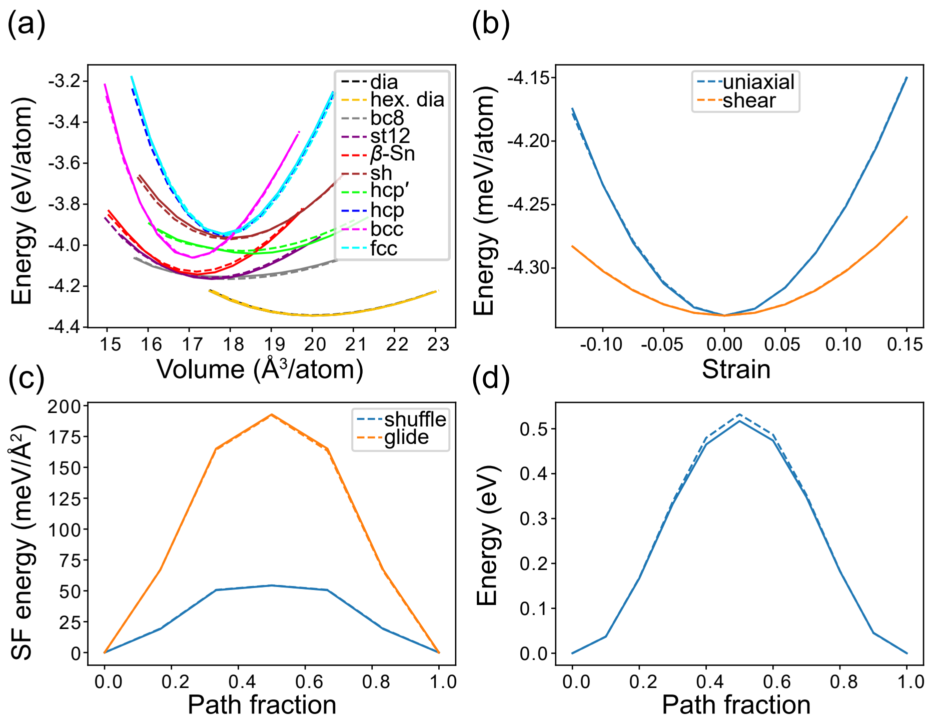


Fig. 4.3: Comparison of various static properties between the reference Stillinger-Weber potential (SW) and neural network potential (NNP) for silicon. Solid lines and dashed lines indicate SW and NNP, respectively. (a) Equation of states for various polymorphs. (b) Energy curve for the uniaxial and shear strain of diamond crystal. (c) Stacking fault energy along the shuffle and glide plane. (d) Vacancy migration energy along the migration path.

(not shown). Therefore, learning Stillinger-Weber potential would be far from trivial.

The example of Stillinger-Weber potential shows that the metadynamics sampling has great potential of sampling a wide range of configurations from a few metadynamics simulations such that it can be used for the development of general-purpose potentials.

Density functional theory

Next, we develop general-purpose potential using DFT calculation as a reference. The target systems are chosen to be silicon and aluminum, typical systems with covalent bonding and metallic bonding, respectively. They are often used as benchmark systems for developing machine learning potentials.

The sampling density of metadynamics and high-temperature molecular dynamics for both silicon (1000–2000 K) and aluminum (300–1500 K) are shown in Fig. 4.4 for silicon and Fig. 4.5 for aluminum. The sampling density is projected onto two principal component axes of $\mathbf{G}$, and it is plotted for two combinations of principal components. It is clearly seen that metadynamics sampling covered a much wider range of $\mathbf{G}$-space for both silicon and aluminum.

For silicon, the same static properties are calculated using the modified version of code from Ref. [93] and listed in Table 4.5. The equation of states for polymorphs, energy-strain curves, energy along stacking fault, and energy along the vacancy migration path is shown in Fig. 4.6. Some properties such as vacancy formation energy, stacking fault energies, vacancy migration barrier, and equation of state for st12, bc8, and hcp$'$ have relatively high error, but most properties are within $\sim$10 % error range. The error level is higher compared to the work by Bartók[93], where the data set was constructed carefully by human intuition. This is the expected result since they included most of the properties in the data set. The neural network potential developed in this work also describes the fourfold defect well, which the GAP developed in Ref. [93] failed to describe. Considering that none of these properties other than elastic constants are explicitly included in the training set, and the training set is generated from just a few metadynamics simulations, it is quite remarkable performance.

The error level is also higher than the classical potential results in Section 4.4.2. It is also expected result for several reasons. First, DFT would have more complex potential energy surface than Stillinger-Weber potential. Therefore, the descriptor vector of higher-resolution or more flexible neural network architecture might be needed.

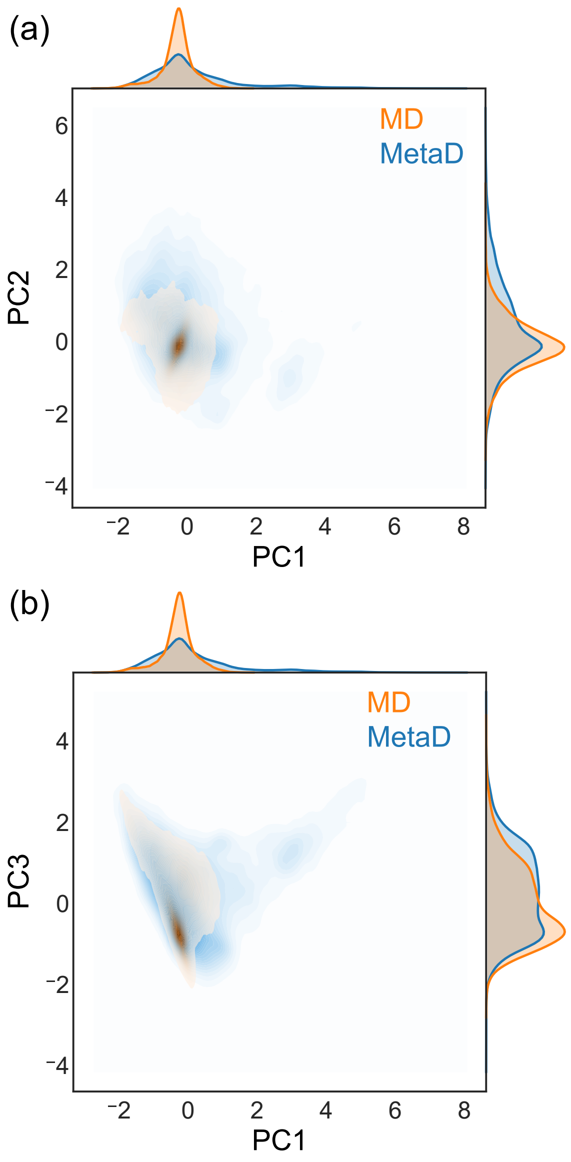


Fig. 4.4: Sampling density of high-temperature molecular dynamics (1000–2000 K; orange) and metadynamics sampling (blue) for silicon. The density is shown in (a) the first and second principal components and (b) the first and third principal components. Additionally, the density projected onto one axis is plotted over the axis. The molecular dynamics simulations start from the diamond crystal structure or diamond crystal (100) slab model.

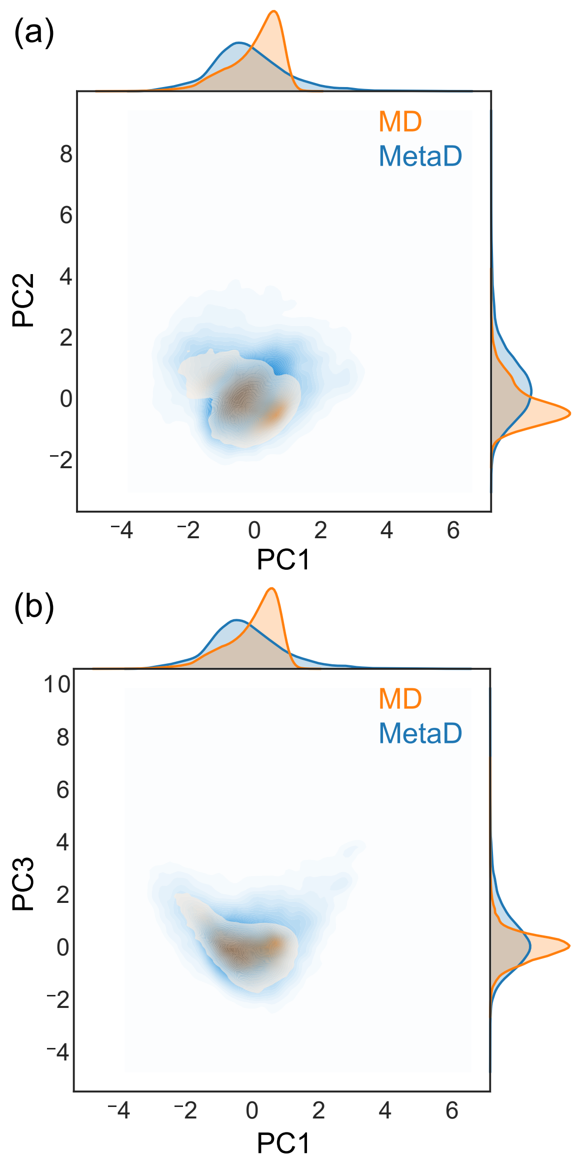


Fig. 4.5: Sampling density of high-temperature molecular dynamics (300–1500 K; orange) and metadynamics sampling (blue) for aluminum. The density is shown in (a) the first and second principal components and (b) the first and third principal components. Additionally, the density projected onto one axis is plotted over the axis. The molecular dynamics simulations start from the same starting structures as metadynamics simulations [FCC, BCC, HCP, FCC(100), and FCC(111) structures].

Table 4.5: Comparison of various static properties between the reference density functional theory (DFT) and the neural network potential (NNP) for silicon. Elastic constants are computed for diamond structure. Surface energies are calculated for (100) (2×2) reconstruction, (110) (1×1) reconstruction, and (111) (3×3) dimer-adatom-stacking-fault (DAS) reconstruction. The abbreviations vac., hex., tetra., db., and ffcd. stand for vacancy, hexagonal interstitial, tetragonal interstitial, dumbbell interstitial, and fourfold coordinated defect, respectively. GB indicates (112) Σ3 grain boundary. $\gamma_{\mathrm{us}}^{(s)}$ and $\gamma_{\mathrm{us}}^{(g)}$ are unstable stacking-fault energies on shuffle plane and glide plane of diamond (111) planes.

		Elastic constant (GPa)				Surf. energy (J/m^2)			Defect formation energy (eV)					Planar def. (J/m^2)			Di-interstitial (eV)				
Model	a_0 (Å)	B	c_{11}	c_{12}	c_{44}	(111)	(110)	(100)	vac.	hex.	tetra.	db.	ffcd.	GB	$\gamma_{\mathrm{us}}^{(s)}$	$\gamma_{\mathrm{us}}^{(g)}$	XX3	EXT	W	XEX	XT
DFT	5.47	89.2	153.7	56.9	74.4	1.24	1.51	1.25	3.64	3.65	3.75	3.60	2.90	0.92	1.57	1.74	5.72	5.73	5.76	5.66	5.37
NNP	5.47	88.9	150.7	58.0	69.0	1.32	1.32	1.25	2.81	3.76	3.33	3.54	2.63	0.91	1.39	2.13	5.53	5.46	5.94	5.73	5.62
%error	0	0	−2	2	−7	6	−13	0	−23	3	−11	−2	−9	−1	−12	23	−3	−5	3	1	5

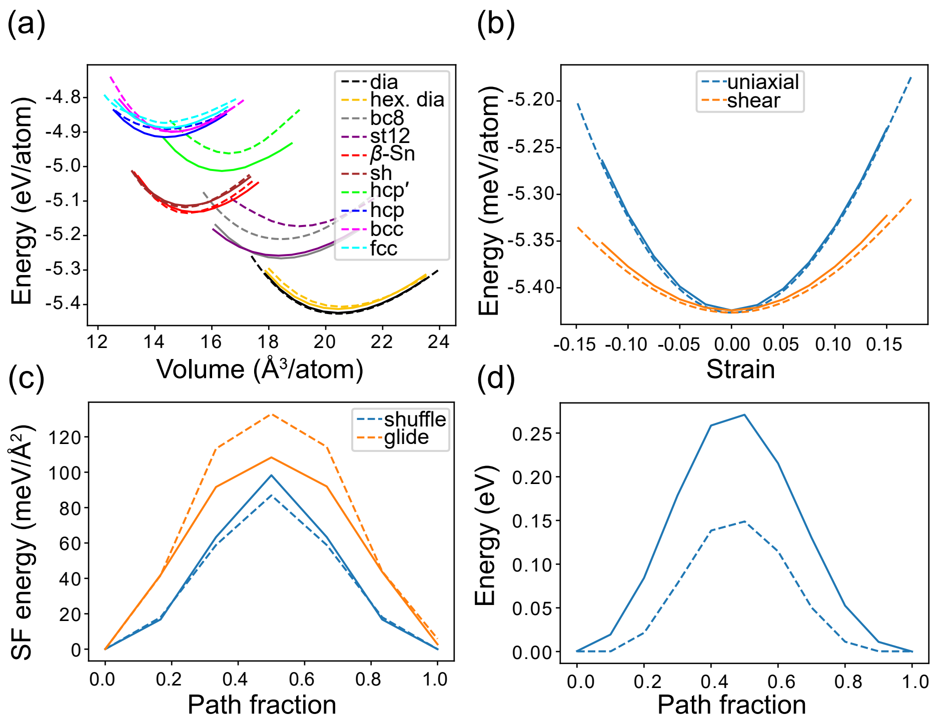


Fig. 4.6: Comparison of various static properties between the reference density functional theory (DFT) and neural network potential (NNP) for silicon. Solid lines and dashed lines indicate DFT and NNP, respectively. (a) Equation of states for various polymorphs. (b) Energy curve for the uniaxial and shear strain of diamond crystal. (c) Stacking fault energy along the shuffle and glide plane. (d) Vacancy migration energy along the migration path.

Due to the complex potential energy surface, NNP can have problems generalizing the structure-energy relationship with the descriptor of low resolution. However, such an increase in the fidelity of the model should be accompanied by the increase of the training set size to avoid overfitting. Second, the error comes from the higher transfer range of DFT. As described in Section 3.2, transferable atomic energy can be defined within DFT, but the transfer range of DFT atomic energy would be much larger than the cutoff distance we use (6.5 Å) for high accuracy. Those errors act as a noise in the data, leading to lower prediction performance of the model. However, increasing the cutoff distance further is prohibiting as the computational cost of neural network potential scales to approximately r_c^6. In order to describe the local environment within a larger cutoff radius, the number of symmetry functions has to be increased too. Therefore, the cost of evaluation would increase more rapidly.

The high error for equation of states of st12 and hcp$'$ structure is expected as the analysis on the distribution of training points shows that those structures are not well covered by the training set. But other properties such as vacancy formation energy appear to be covered by the training set. Therefore, those errors are likely from the reasons described above and not from the lack of sampling.

As another validation, we performed dynamics simulations with NNP and calculated DFT energy on the snapshots along the trajectory. The results are shown in Fig. 4.7 for silicon and Fig. 4.8 for aluminum. Four trajectories encompassing various structures are generated with silicon NNP; a melt-quench trajectory, a heating (melting) trajectory of a nanocluster, a metadynamics trajectory, and a heating trajectory of silicene. Similarly, four trajectories are generated for aluminum; a melt-quench trajectory, a heating (melting) trajectory of a nanocluster, and two metadynamics trajectories. For the trajectories encompassing various structures, NNP showed outstanding energy correlation with DFT, besides shifts for some trajectories. The shift in energy is observed for systems with surfaces and high energy structures generated with metadynamics. The error in surface energy or the energy of corner atoms can result a constant shift in energy. They affect the formation energy of the system, but the dynamics of the system would not be affected. Also, during the metadynamics simulation, voids are formed inside the bulk. Therefore, the energy shift grows during the simulation as the void grows. Other than the energy shift, NNPs trained with metadynamics training set show outstanding energy correlation to the reference without any failures. The results show that the NNPs can describe such various systems well. On the other hand, the same test on NNPs that are trained with high-temperature MD results in failures, and the energy correlation is lost, indicating that the metadynamics sampling is superior to molecular dynamics sampling.

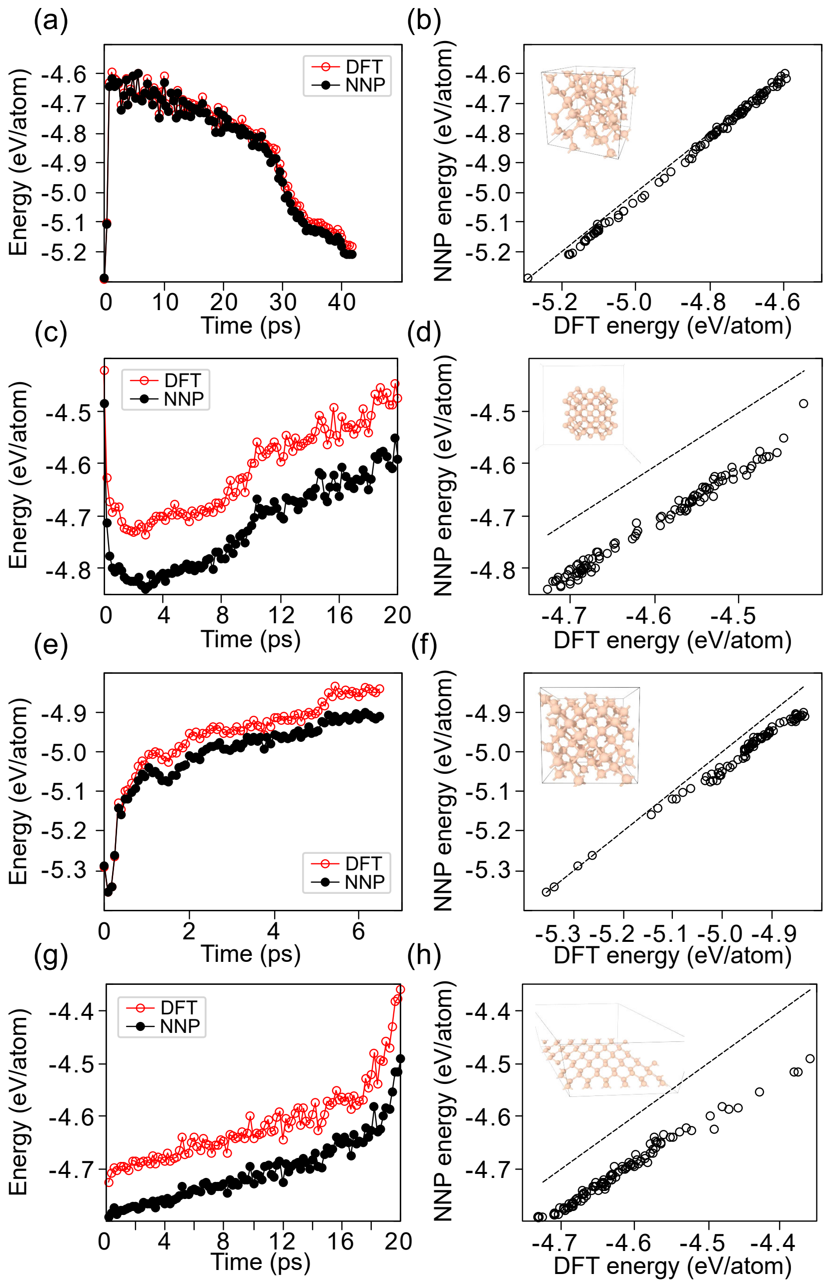


Fig. 4.7: Energy correlation between density functional theory (DFT) and neural network potential (NNP) for silicon on various trajectories generated by NNP. (a,c,e,g) Energy of DFT and NNP along the trajectory. (b,d,f,h) Correlation between DFT energy and NNP energy. Dotted line is $y = x$ line. Initial structures are shown as insets inside the correlation plot.

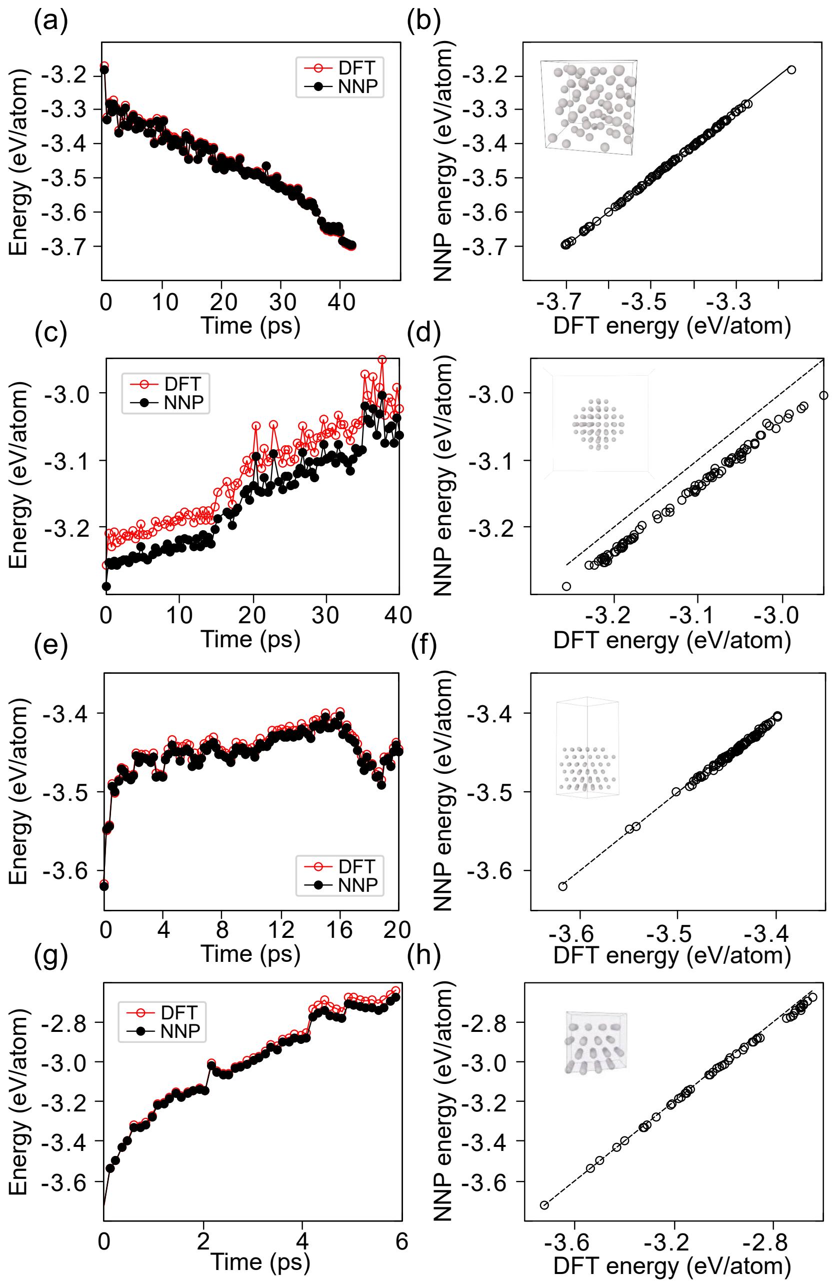


Fig. 4.8: Energy correlation between density functional theory (DFT) and neural network potential (NNP) for aluminum on various trajectories generated by NNP. (a,c,e,g) Energy of DFT and NNP along the trajectory. (b,d,f,h) Correlation between DFT energy and NNP energy. Dotted line is $y = x$ line. Initial structures are shown as insets inside the correlation plot.

4.4.3 Improving stability of potentials

The last application is to generate a robust training set, which improves the stability of machine learning potentials. This is closely related to the first application of catching unexpected failures. Since metadynamics sampling searches local configuration space from the low energy to higher energy, it can sample higher energy states and barriers so that machine learning potential does not have a hole in potential energy surface up to a certain energy level.

We demonstrate that metadynamics sampling results in more robust potentials by comparing neural network potentials trained in Section 4.4.2. In this section, we denote the neural network potential trained with metadynamics training set by NNP-MetaD and the one trained with conventional high-temperature molecular dynamics training set by NNP-MD. Although molecular dynamics training set includes high-temperature configurations and thus high energy states, one could find possible failures of NNP-MD easily by performing metadynamics simulation (see the first application). The results are shown in Fig. 4.9.

For silicon potential, the energy of NNP-MD remained at the almost same level during the formation of short triangles [which is made of bonds shorter than 2 Å; see Fig. 4.9(b)]. The number of short triangles are plotted as a green line in Fig. 4.9(a). DFT energy on NNP-MD trajectory confirms that it is actually high energy structure, and NNP-MD failed. NNP-MetaD energy on the same trajectory similarly gives high energy, implying the higher stability of NNP-MetaD. The same metadynamics protocol was applied for NNP-MetaD, and the results are shown in Fig. 4.10(a). Other than slowly increasing energy shift, NNP-MetaD energy and DFT energy shows good correlation without a sudden increase in the gap. The energy shift is due to the underestimation of surface energy, and the shift increases as the voids form and grow. The results in Fig. 4.10 confirms the higher stability of NNP-MetaD than NNP-MD.

The same story applies to the aluminum potential. The energy of NNP-MD decreases during the formation of short bonds [bond length of 1.8–2.3 Å; see Fig. 4.9(d)], whose number is plotted in the figure as a green line in Fig. 4.9(c). However, when we calculate DFT energy along those trajectories, we can clearly see that NNP-MD is in fact going to high energy configurations, indicating a failure. The blue circles show the energy calculated with NNP-MetaD on the same trajectory. It can be seen that NNP-MetaD gives a good description of the high energy region. This implies that NNP-MetaD would not fail in the same circumstances. Hence NNP-MetaD has higher stability than NNP-MD. For a fair comparison, we again did a metadynamics simulation with NNP-MetaD with the same protocol. The results are shown in Fig. 4.10(b). NNP-MetaD did not show any sign of failure during the same simulation protocol, which confirms the higher stability of NNP-MetaD.

Although not demonstrated here, metadynamics sampling is expected to improve stability for multi-component systems too. For example, the phase separation problem of GeTe potential would be solved as metadynamics automatically searches over the composition space.

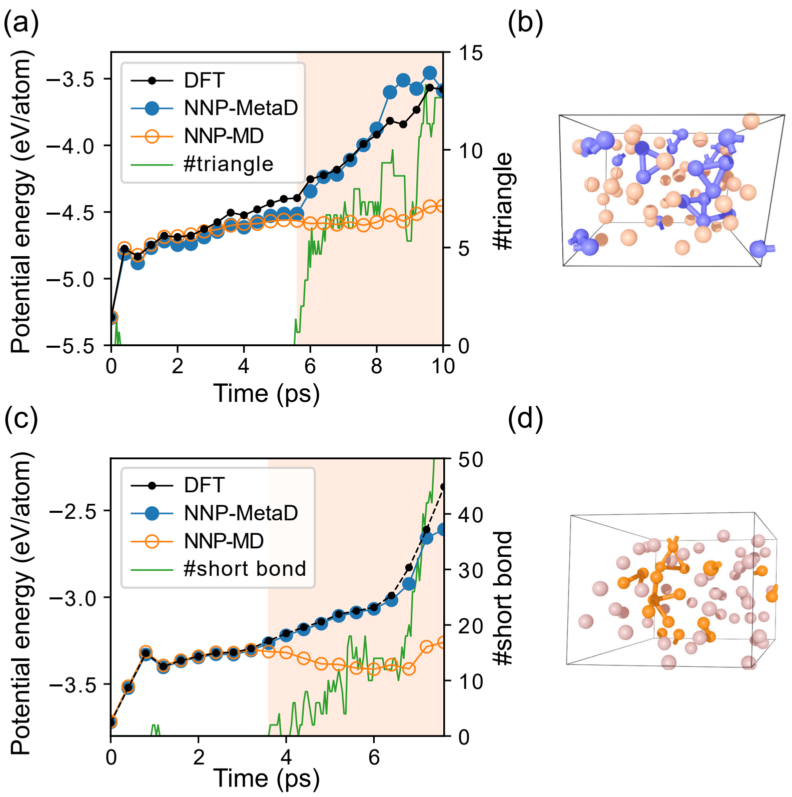


Fig. 4.9: Stability test results with NNP-MD. (a) The potential energy of DFT, NNP-MetaD, and NNP-MD along the metadynamics trajectory of NNP-MD for silicon. The number of triangles with short bonds (< 2 Å) is given on the right axis, and the corresponding high-energy structure is shown in (b) with the triangles highlighted in blue. (c) The potential energy of DFT, NNP-MetaD, and NNP-MD along the metadynamics trajectory of NNP-MD for aluminum. The number of short bonds (1.8–2.3 Å) is plotted in the right axis. The high-energy structure is shown in (d) with short bonds highlighted in orange.

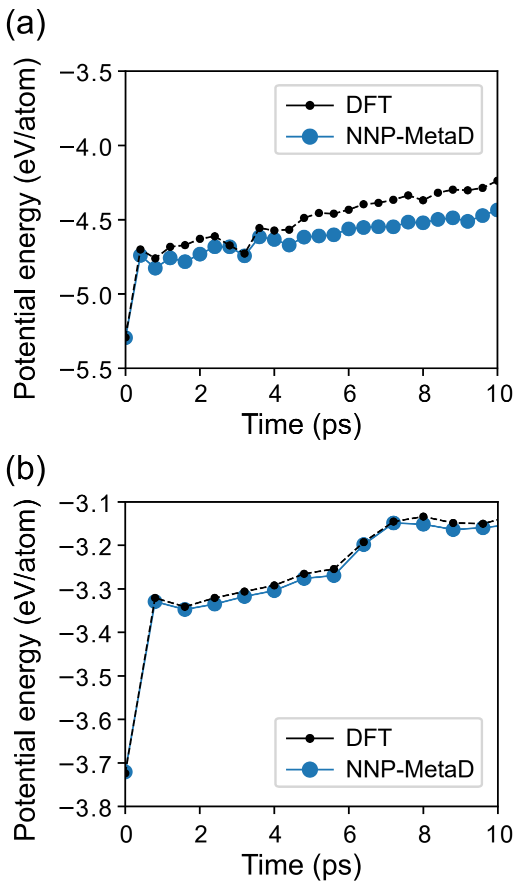


Fig. 4.10: Stability test results with NNP-MetaD. The protocol is identical to the results in Fig. 4.9. The potential energy of DFT and NNP-MetaD along the metadynamics trajectory of NNP-MetaD for (a) silicon and (b) aluminum.

4.5 Summary

In summary, we suggested a novel metadynamics scheme, where the input descriptor is directly used as a collective variable, and the total bias potential is given as a sum of atomic bias potentials. In the suggested scheme, the simulation is biased to search local environment space of each atom simultaneously. Three applications of suggested metadynamics for the development of machine learning potentials are demonstrated. First, it can be used to catch possible failures of premade machine learning potentials efficiently. Second, diverse configurations can be generated from metadynamics simulations so that they can be used to develop general-purpose potentials. Lastly, it was shown that the training set generated with the metadynamics results in more robust potentials. These applications represent major bottlenecks to the development of robust machine learning potentials. Therefore, the proposed metadynamics scheme would contribute to the community by greatly easing and accelerating the development process of machine learning potentials.

Chapter 5

Conclusion

We addressed two issues that are major hurdles to the development of machine learning potentials. First, we rigorously defined the expression for transferable atomic energy within DFT, paying attention to its transfer range. The existence of transferable atomic energy implies that the objective of machine learning potential is to infer the underlying atomic energy function when only total energies are informed. We showed that machine learning potentials have the capability to infer the underlying atomic energy function from total energies. On the other hand, it was also shown that machine learning potentials can give accurate total energy predictions for the training set but give markedly wrong atomic energy due to the unconventional structure of learning problem. This ad hoc energy mapping occurs due to the limitation in the training set, such as fixed volume, fixed composition, and a fixed number of atoms in each distinct configuration. Care must be taken as the ad hoc energy mapping undermines the transferability of machine learning potentials, and it becomes difficult to prevent ad hoc energy mapping by a deliberate choice of a training set when it comes to complex systems. Moreover, ad hoc energy mapping can be unnoticed by the conventional monitoring process because a machine learning potential gives accurate total energies and atomic forces for the training set regardless of ad hoc energy mapping. By elucidating the learning problem of machine learning potential, the present work would contribute to the development of transferable machine learning potential. Next, we suggested a new form of metadynamics where the local environment descriptor vector is used as a collective variable, and the bias potential is a sum of atomic bias potentials. The method directly enhances sampling over local environment space such that a general-purpose potential can be easily developed. The general-purpose potential for silicon and aluminum were developed by using metadynamics sampling for demonstration. Another important aspect is that metadynamics gives a robust training set, which improves the stability of a machine learning potential. It was shown that the metadynamics sampling results in the potential with higher stability compared to high-temperature molecular dynamics sampling. Furthermore, it can be used to measure the stability of premade potentials such that the stability issues can be detected earlier. By allowing easier construction of a robust training set covering a wide range of configurations, the metadynamics sampling method suggested in the dissertation would accelerate the development process of machine learning potentials, and thus contribute to the wider applications of machine learning potentials. As a final remark, we mention the current challenges of machine learning potentials briefly. Although machine learning potentials are being successfully employed to complicated systems, opening new possibilities, developing high-quality machine learning potential is still a challenging task. We need to deepen the fundamental understanding of machine learning potential to navigate through the problems that arise during development. Also, it is always of question that the simulation results are reliable or not due to the black-box nature of machine learning. Therefore, reliable uncertainty quantification methods are essential for machine learning potential to be widely adopted and become a standard tool for atomistic simulations.

Bibliography

[1] J. Behler and M. Parrinello, Phys. Rev. Lett. **98**, 146401 (2007).

[2] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, Phys. Rev. Lett. **104**, 136403 (2010).

[3] F. C. Mocanu, K. Konstantinou, T. H. Lee, N. Bernstein, V. L. Deringer, G. Csányi, and S. R. Elliott, J. Phys. Chem. B **122**, 8998 (2018).

[4] M. Hellström, V. Quaranta, and J. Behler, Chem. Sci. **10**, 1232 (2019).

[5] G. Sun and P. Sautet, J. Am. Chem. Soc. **140**, 2812 (2018).

[6] Z. W. Ulissi, M. T. Tang, J. Xiao, X. Liu, D. A. Torelli, M. Karamad, K. Cummins, C. Hahn, N. S. Lewis, T. F. Jaramillo, et al., ACS Catal. **7**, 6600 (2017).

[7] W. Li and Y. Ando, Phys. Chem. Chem. Phys. **20**, 30006 (2018).

[8] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand, J. Chem. Phys. **148**, 241709 (2018).

[9] G. Imbalzano, A. Anelli, D. Giofré, S. Klees, J. Behler, and M. Ceriotti, J. Chem. Phys. **148**, 241730 (2018).

[10] S. Hajinazar, J. Shao, and A. N. Kolmogorov, Phys. Rev. B **95**, 014114 (2017).

[11] B. Onat, E. D. Cubuk, B. D. Malone, and E. Kaxiras, Phys. Rev. B **97**, 094106 (2018).

[12] T. L. Pham, H. Kino, K. Terakura, T. Miyake, and H. C. Dam, J. Chem. Phys. **145**, 154103 (2016).

[13] W. Jeong, K. Lee, D. Yoo, D. Lee, and S. Han, J. Phys. Chem. C **122**, 22790 (2018).

[14] A. Thompson, L. Swiler, C. Trott, S. Foiles, and G. Tucker, J. Comput. Phys. **285**, 316 (2015).

[15] A. V. Shapeev, Multiscale Model. Simul. **14**, 1153 (2016).

[16] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, Nat. Commun. **8**, 13890 (2017).

[17] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, Sci. Adv. **3** (2017).

[18] M. Born and R. Oppenheimer, Ann. Phys. (Berl.) **389**, 457 (1927).

[19] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[20] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[21] J. P. Perdew and W. Yue, Phys. Rev. B **33**, 8800 (1986).

[22] A. D. Becke, Phys. Rev. A **38**, 3098 (1988).

[23] W. Kohn, Phys. Rev. Lett. **76**, 3168 (1996).

[24] E. Prodan and W. Kohn, Proc. Natl. Acad. Sci. U.S.A. **102**, 11635 (2005).

[25] S. Goedecker, Rev. Mod. Phys. **71**, 1085 (1999).

[26] W. Yang, Phys. Rev. Lett. **66**, 1438 (1991).

[27] W. Yang and T. Lee, J. Chem. Phys. **103**, 5674 (1995).

[28] J. S. Smith, O. Isayev, and A. E. Roitberg, Chem. Sci. **8**, 3192 (2017).

[29] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, J. Chem. Phys. **148**, 241722 (2018).

[30] L. Zhang, J. Han, H. Wang, R. Car, and W. E, Phys. Rev. Lett. **120**, 143001 (2018).

[31] K. Hornik, Neural Netw. **4**, 251 (1991).

[32] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, J. Chem. Phys. **103**, 4129 (1995).

[33] H. Gassner, M. Probst, A. Lauenstein, and K. Hermansson, J. Phys. Chem. A **102**, 4596 (1998).

[34] S. Lorenz, A. Groß, and M. Scheffler, Chem. Phys. Lett. **395**, 210 (2004).

[35] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms* (Cambridge university press, 2003).

[36] J. Behler, J. Chem. Phys. **134**, 074106 (2011).

[37] A. P. Bartók, R. Kondor, and G. Csányi, Phys. Rev. B **87**, 184115 (2013).

[38] G. Imbalzano, A. Anelli, D. Giofré, S. Klees, J. Behler, and M. Ceriotti, J. Chem. Phys. **148**, 241730 (2018).

[39] M. W. Mahoney and P. Drineas, Proc. Natl. Acad. Sci. U.S.A. **106**, 697 (2009).

[40] S. Wold, K. Esbensen, and P. Geladi, Chemom. Intell. Lab. Syst. **2**, 37 (1987).

[41] M. S. Advani and A. M. Saxe, arXiv e-prints arXiv:1710.03667 (2017).

[42] J. Behler, Int. J. Quantum Chem. **115**, 1032 (2015).

[43] R. Lot, F. Pellegrini, Y. Shaidu, and E. Kucukbenli, arXiv e-prints arXiv:1907.03055 (2019).

[44] N. Artrith and A. Urban, Comput. Mater. Sci. **114**, 135 (2016).

[45] A. Khorshidi and A. A. Peterson, Comput. Phys. Commun. **207**, 310 (2016).

[46] A. S. Bochkarev, A. van Roekeghem, S. Mossa, and N. Mingo, Phys. Rev. Materials **3**, 093803 (2019).

[47] K. Lee, D. Yoo, W. Jeong, and S. Han, Comput. Phys. Commun. **242**, 95 (2019).

[48] K. He, X. Zhang, S. Ren, and J. Sun, arXiv e-prints arXiv:1502.01852 (2015).

[49] J. Duchi, E. Hazan, and Y. Singer, J. Mach. Learn. Res. **12**, 2121 (2011).

[50] D. P. Kingma and J. Ba, arXiv e-prints arXiv:1412.6980 (2014).

[51] M. Gastegger and P. Marquetand, J. Chem. Theory Comput. **11**, 2187 (2015).

[52] Y. Huang, J. Kang, W. A. Goddard, and L.-W. Wang, Phys. Rev. B **99**, 064103 (2019).

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014).

[54] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, arXiv e-prints arXiv:1611.03530 (2016).

[55] A. Laio and M. Parrinello, Proc. Natl. Acad. Sci. U.S.A. **99**, 12562 (2002).

[56] G. Torrie and J. Valleau, J. Comput. Phys. **23**, 187 (1977).

[57] E. Darve and A. Pohorille, J. Chem. Phys. **115**, 9169 (2001).

[58] A. Barducci, G. Bussi, and M. Parrinello, Phys. Rev. Lett. **100**, 020603 (2008).

[59] C. Domene, P. Barbini, and S. Furini, J. Chem. Theory Comput. **11**, 1896 (2015).

[60] J. Pfaendtner and M. Bonomi, J. Chem. Theory Comput. **11**, 5062 (2015).

[61] A. Prakash, C. D. Fu, M. Bonomi, and J. Pfaendtner, J. Chem. Theory Comput. **14**, 4985 (2018).

[62] D. Branduardi, G. Bussi, and M. Parrinello, J. Chem. Theory Comput. **8**, 2247 (2012).

[63] M. A. Brubaker, A. Geiger, and R. Urtasun, IEEE Trans. Pattern Anal. Mach. Intell. **38**, 652 (2016).

[64] N. Vasconcelos and A. Lippman, in *Advances in Neural Information Processing Systems* (1999), pp. 606–612.

[65] K. Zhang and J. T. Kwok, IEEE Trans. Neural Netw. **21**, 644 (2010).

[66] L. Yu, T. Yang, and A. B. Chan, IEEE Trans. Pattern Anal. Mach. Intell. **41**, 1323 (2019).

[67] M. Yu, D. R. Trinkle, and R. M. Martin, Phys. Rev. B **83**, 115113 (2011).

[68] P. L. A. Popelier, Int. J. Quantum Chem. **115**, 1005 (2015).

[69] G. C. Sosso, G. Miceli, S. Caravati, F. Giberti, J. Behler, and M. Bernasconi, J. Phys. Chem. Lett. **4**, 4241 (2013).

[70] N. Artrith, T. Morawietz, and J. Behler, Phys. Rev. B **83**, 153101 (2011).

[71] T. Morawietz, V. Sharma, and J. Behler, J. Chem. Phys. **136**, 064103 (2012).

[72] J. Behler, Angew. Chem. Int. Ed. **56**, 12828 (2017).

[73] Y. Huang, J. Kang, W. A. Goddard, and L.-W. Wang, Phys. Rev. B **99**, 064103 (2019).

[74] N. Chetty and R. M. Martin, Phys. Rev. B **45**, 6074 (1992).

[75] J. Cai and Y. Y. Ye, Phys. Rev. B **54**, 8398 (1996).

[76] N. Artrith and J. Behler, Phys. Rev. B **85**, 045439 (2012).

[77] A. A. Peterson, R. Christensen, and A. Khorshidi, Phys. Chem. Chem. Phys. **19**, 10978 (2017).

[78] E. V. Podryabinkin and A. V. Shapeev, Comput. Mater. Sci. **140**, 171 (2017).

[79] L. Zhang, D.-Y. Lin, H. Wang, R. Car, and W. E, Phys. Rev. Materials **3**, 023804 (2019).

[80] R. Jinnouchi, F. Karsai, and G. Kresse, Phys. Rev. B **100**, 014105 (2019).

[81] V. L. Deringer, C. J. Pickard, and G. Csányi, Phys. Rev. Lett. **120**, 156001 (2018).

[82] J. E. Herr, K. Yao, R. McIntyre, D. W. Toth, and J. Parkhill, J. Chem. Phys. **148**, 241710 (2018).

[83] C. J. Pickard and R. J. Needs, J. Phys. Condens. Matter **23**, 053201 (2011).

[84] N. Bernstein, G. Csányi, and V. L. Deringer, npj Comput. Mater. **5**, 99 (2019).

[85] S. Plimpton, J. Comput. Phys. **117**, 1 (1995).

[86] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, et al., J. Phys. Condens. Matter **29**, 273002 (2017).

[87] G. Kresse and J. Hafner, Phys. Rev. B **49**, 14251 (1994).

[88] G. Kresse and J. Furthmüller, Comput. Mater. Sci. **6**, 15 (1996).

[89] G. Kresse and J. Furthmüller, Phys. Rev. B **54**, 11169 (1996).

[90] G. Kresse and D. Joubert, Phys. Rev. B **59**, 1758 (1999).

[91] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

[92] V. Botu, R. Batra, J. Chapman, and R. Ramprasad, J. Phys. Chem. C **121**, 511 (2017).

[93] A. P. Bartók, J. Kermode, N. Bernstein, and G. Csányi, Phys. Rev. X **8**, 041048 (2018).

[94] F. H. Stillinger and T. A. Weber, Phys. Rev. B **31**, 5262 (1985).

초 록

기계 학습 퍼텐셜은 제일원리 계산에 근접하는 정확도를 훨씬 적은 계산 비용으로 계산하는 방법으로 주목받고 있다. 특히 계산 비용이 원자 수에 비례하여 증가하기 때문에 높은 정확도로 대규모 계산을 수행하기에 적합한 방법이다. 복잡한 계에 대한 기존 응용 연구들이 성공적으로 수행되어 기계 학습 퍼텐셜의 가능성을 보여주었다. 하지만 기계 학습 퍼텐셜에 대한 기초적인 이해가 부족하고 학습 세트를 선정하는 것이 어렵기 때문에 아직 널리 사용되지 못하고 있다. 기계 학습 퍼텐셜과 관련된 방법론은 빠르게 발전하고 있지만, 기계 학습 모델이 블랙박스처럼 사용되어 여전히 기초적인 이해가 부족한 실정이다. 또한 밀도 범함수 이론 결과를 학습하는 기계 학습 퍼텐셜이 성립하려면 기계 학습 모델과 밀도 범함수 이론의 관계가 중요한데 이에 관한 내용은 아직 논의되지 않았다. 다른 한편으로는 높은 신뢰도를 가진 기계 학습 퍼텐셜을 개발하려면 신중하게 학습 세트를 선정하는 것이 중요하다. 학습 세트는 보통 직관과 경험, 그리고 시행착오를 통해 구성하는데 여전히 시뮬레이션 도중에 예상치 못한 구조가 발생하며 시뮬레이션이 실패하는 경우가 흔히 발생한다. 따라서 쉽고 체계적으로 학습 세트를 구축하는 방법이 요구된다.

이 연구에서는 기계 학습 퍼텐셜이 널리 사용되는 데 방해가 되는 두 가지 문제점에 대해 논의한다. 먼저, 전자 구조의 국지성 원리로부터 시작하여 밀도 범함수 이론에서 양도 가능한 원자 에너지를 정의하였다. 이는 국지성에 기반을 둔 모든 기계 학습 퍼텐셜에 있어서 필수적으로 정의되어야 한다. 이로부터 기계 학습 퍼텐셜은 전체 에너지로부터 기저에 있는 원자 에너지 함수를 배우는 것임을 알 수 있다. 고전 퍼텐셜을 이용한 예시를 통해 기계 학습 퍼텐셜이 실제로 전체 에너지만이 주어졌을 때 기저에 있는 원자 에너지 함수를 배울 수 있음을 보였다. 또한, 세 가지 예시를 통해 기계 학습 퍼텐셜이 전체 에너지는 정확히 맞추지만 잘못된 원자 에너지를 배울 수 있음을 보였다. 잘못된 원자 에너지가 다성분계에 미치는 영향도 논의하였다. 다음으로 넓은 원자 환경을 샘플링하여 학습 세트를 쉽게 구축할 수 있는 새로운 메타동역학 샘플링 기법을 제안하였다. 제안된 방법에서 집합 변수(collective variable)

로는 기계학습에 사용되는 디스크립터 벡터를 사용하고 전체 바이어스 퍼텐셜은 개별 원자의 바이어스 퍼텐셜들의 합으로 주어진다. 이렇게 함으로써 전체 계의 구조 공간을 탐색하기보다는 개별 원자의 주변 환경 공간을 효율적으로 탐색할 수 있다. 이 연구에서는 제안된 메타동역학 방법의 세 가지 응용 예시를 보였다. 먼저, 메타동역학 샘플링 기법을 활용하여 실리콘과 알루미늄에 대해 광범위하게 사용될 수 있는 다목적 퍼텐셜을 개발하였다. 또한, 메타동역학 방법이 이미 만들어진 퍼텐셜의 안정성을 평가하는 데 사용될 수 있음을 보이고, 메타동역학 샘플링 기법이 퍼텐셜의 안정성을 향상하는 것을 보였다.

주요어: 기계 학습 퍼텐셜, 인공 신경망 퍼텐셜, 메타동역학
학번: 2015-20848