Machine Learning Models and Uncertainty for Atomic Simulations

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

Department of Chemical Engineering

Ni Zhan

B.S., Chemical Engineering, The University of Texas at Austin
M.S., Machine Learning, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, Pennsylvania

December, 2021

© Ni Zhan, 2021

All Rights Reserved

Acknowledgments

First, I would like to thank my advisor Professor John Kitchin. I learned so much scientific and research knowledge from John. He encouraged me to persevere on problems when I could barely see a path forward, and inspired me to try new and interesting research directions. I thank him for support throughout my PhD and being especially understanding during unique situations.

I want to thank my PhD committee, Prof. Zachary Ulissi, Prof. Michael Widom, Prof. Aditya Khair, and Prof. Erik Ydstie, for their time and insight. They are inspirational to me, and I am grateful and honored for the opportunity to discuss research with them. I want to especially thank Prof. Widom for the time we attended the High Performance Computing conference and collaboration on the Ni-Al-W work.

Additionally on the Ni-Al-W work, I would like to thank Dr. Jim Lill, Dr. Chris Woodward, and Dr. Bojun Feng for their collaboration. I also want to thank Prof. Alan McGaughey for discussion and feedback on Al-Si work.

I gratefully acknowledge funding support from Project PP-CCM-KY09-002-P3, Complex Atomistic Potentials through Machine Learning of the User Productivity Enhancement, Technology Transfer, and Training (PETTT) Program, the Department of Chemical Engineering, H. Robert Sharbaugh Presidential Fellowship, and Thomas and Adrienne Klopack Fellowship.

I want to thank my mentors from undergraduate study, Prof. Wei Li, Prof. Thomas Edison, Prof. Amir Mohammadi, Prof. Michael Baldea, Prof. Juan Ruiz, Dr. Hao Xin, Dr. Rohit Jain, and Dr. Jingyu Ock. Thank you for believing in me, and thanks to all of the educators and professors who have helped me.

Thanks to Mingjie Liu, Yilin Yang, Maya Bhat, Junwoong Yoon, Dr. Yijia Sun, and Jie Gong for helpful discussions and collaborative projects. I thank the friends I met in graduate school for their support and shared experiences, and thank all my friends for moral support.

Finally, I thank my parents for their love and support, and my mom for encouraging me, to pursue the PhD and always.

Abstract

As computational power grows, materials simulation becomes an increasingly valuable scientific tool. Simulations are used to calculate properties that are difficult to obtain experimentally, perform high-throughput design and discovery, and investigate material behavior and theory. Ultimately we want to push the time and length scales of simulation and connect atomic scale with continuum scale properties. There is a trade-off between accuracy and computational cost. One approach is to use machine learned (ML) potentials as fast, accurate approximations to quantum chemical methods. ML potentials are systematically improvable, and can be as accurate as density functional theory at much lower computational cost. Potential challenges are that ML potentials extrapolate, and it is nonobvious when extrapolation occurs and how to efficiently build the training dataset. In addition we would like to have uncertainty measurements of ML models and physically simulated properties.

In this dissertation, we investigate the relationship between atomic scale and continuum properties in liquid Al-Si using molecular dynamics (MD). We study the local order using Voronoi polyhedrons and agglomerative clustering, which allowed us to analyze the large amount of data generated from MD trajectories in an efficient manner. We found that clusters have minimal effect on diffusion while increasing viscosity, which is a likely origin of the Stokes-Einstein deviation for liquid Al-Si at low temperatures near the melting point. This study demonstrates the value of MD simulation and using ML clustering and large datasets analysis to find new phenomena.

In the next part, we build a neural network (NN) potential for a complex Ni-Al-W liquid alloy. We conducted hyperparameter studies on NN

architecture and Behler-Parrinello fingerprints. The ML potential was iteratively tested in MD simulation and retrained with diverse dataset. The final potential achieved comparable results with *ab initio* simulation. We found that the NN potential extrapolated on inputs that were dissimilar from its training data, which motivates uncertainty quantification.

We implement the multiparameter delta method for NN potentials (and generally for other nonlinear models) with parameters trained by least squares regression. The uncertainty measure requires the gradient of the model prediction and the Hessian of the loss function, both with respect to model parameters. We obtain the derivatives from ML software with automatic differentiation. We show that the uncertainty measure is larger for input space regions that are not part of the training data. Therefore this method can be used to identify extrapolation, aid in selecting training data, and assess model reliability.

In the final part, we compare uncertainties of physical properties from delta method, Bayesian nonlinear regression, and Gaussian process (GP). Many physical properties of interest require derivatives, therefore we derived GP with joint distribution over a function and its first and second derivatives. We show that delta method and Bayesian nonlinear regression give model specific uncertainty while GP variance includes uncertainty with respect to model selection.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Materials simulation becomes a further powerful tool with ever increasing computational power. Simulations are used to calculate material properties which are difficult to measure experimentally,[1–3] and they aid discovery of new catalysts, drugs, and other materials with target properties by screening an enormous search space.[4–8] Scale of simulations ranges from atomic level to macroscopic behaviors; with advances in computational power and modeling techniques, we work toward bridging these scales.

As we make scientific progress in simulations, there are also important challenges. At the highest level accuracy, simulations are limited to length scale of nanometers.[9] With increasing use of simulations and computer storage, large amounts of data are generated, and we must efficiently extract their contained information. The derivation of properties from materials modeling also begets the question of uncertainty in their calculation.

In this dissertation, we explore the use of machine learning to address these challenges. We show how observed phenomena in atomic simulation may explain observable continuum properties. The main results are obtained using clustering techniques and analyzing large datasets of atomic configurations. We develop a machine learned potential to accelerate simulations, integrate it with molecular dynamics, and discuss the associated opportunities and challenges. Finally we discuss model uncertainty quantification for parameterized potentials and generally nonlinear regression models, and Bayesian modeling to obtain uncertainties of physical properties. The remaining chapter discusses related technical background and organization of dissertation.

1.2 Technical background

1.2.1 Potential energy surface

The potential energy surface (PES) is a central concept in computational chemistry, and is used to explore atomic structures and chemical reactions.[10] The PES is the mapping between the energy and geometry of a system of atoms. For a geometry of the atomic system $\mathbf{r}$, the energy $U(\mathbf{r})$ is the PES. We also define the forces as the derivative $-\frac{\partial U}{\partial \mathbf{r}}$.

In the simple case of interaction between two atoms, we can express the geometry in one dimension as the interatomic distance, as shown in Fig. 1.1. In another example of face-centered cubic (FCC) Au, we can dimensionally reduce its geometric definition from the Cartesian coordinates to its volume/atom. The PES for the FCC Au system is also an equation of state and determines the equilibrium lattice constant. We can often reduce dimensionality of geometric configuration; another example is O-H bond length and H-O-H bond angle to fully specify structure of water molecule. However in many systems it is non-obvious how the Cartesian coordinates can be dimensionally reduced, such as the case for the liquid alloy system shown.

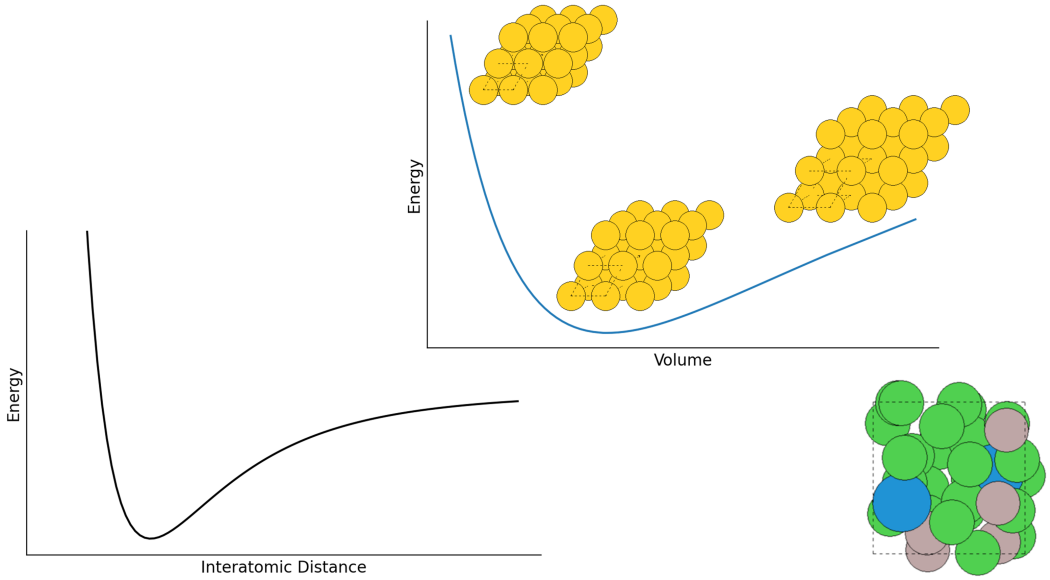


Figure 1.1: Examples of PES, in one dimension, dimensionally reduced system, and liquid alloy system, a high dimensional system for which dimension reduction from Cartesian coordinates is less obvious.

There are different methods to obtain the energy for an atomic configuration. The *ab initio* method density functional theory (DFT) aims to determine ground-state energy from the Schrödinger equation. DFT is versatile, commonly used in physics, chemistry, and materials science, and considered highly accurate among computational approaches.[11] A major limitation is high computational cost. Physical potentials are based on physics approximations and have a functional form with parameters to be fitted, usually to some combination of experimental and/or *ab initio* data. As examples, an embedded atom method potential for Ni, Pd fitted functions of the potential to experimental values including lattice constant and elastic constants,[12] a Reaxff potential for hydrocarbons fitted parameters to their heats of formation and molecular geometries,[13] and angular-dependent potential fitted parameters to *ab initio* forces[14,15] and another to combination of experimental and *ab initio* data.[16] The accuracy of the physical potential at different conditions or on different properties than those

used in the fitting process (transferability) is usually not known beforehand. Physical potentials are also difficult to systematically improve.[17] Machine learned potentials have highly flexible functional forms and are the most data-driven potential type. We discuss machine learned potentials further in Chapter 2.

1.2.2 Molecular dynamics simulation

Molecular dynamics (MD) is a simulation method for physical movement of atoms and molecules. The method dynamically progresses a system of particles following equations of motion (differential equations) by numerical methods. MD generates microscopic information (particle positions and velocities), and the collection of microscopic states follow a statistical ensemble from statistical mechanics. The microscopic state (microstate) belongs to a phase space which has $6N$ dimensions for a system with N particles. The statistical ensemble is the probability distribution for the states of the system. Tuckerman et al. have developed a framework for designing MD algorithms to replicate the desired statistical ensemble.[18] Following this, we can obtain some macroscopic property $\mathcal{A}$ (for example potential energy) as an ensemble average. For our observed property $\mathcal{A}_{\text{obs}}$

$$\mathcal{A}_{\text{obs}} = \langle \mathcal{A} \rangle_{\text{ens}} = \int_{\Gamma} \mathcal{A}(\Gamma) \rho_{\text{ens}}(\Gamma) d\Gamma \tag{1.1}$$

where $\langle \mathcal{A} \rangle_{\text{ens}}$ is the ensemble average, Γ is the microstate, and ρ_{ens} is the probability density of the ensemble.[19] We usually assume that the simulation

was run long enough and is able to pass through states at their ensemble probability (ergodic hypothesis) such that

$$\langle \mathcal{A} \rangle_{\text{ens}} = \langle \mathcal{A} \rangle_{\text{time}} = \frac{1}{T} \sum_{\tau=1}^{T} \mathcal{A}(\Gamma(\tau)) \tag{1.2}$$

where $\tau = 1, ..., T$ represents index of times.

The popular ensembles used for MD simulation are microcanonical (NVE), canonical (NVT), and isothermal-isobaric (NPT). In the microcanonical ensemble, the system volume, Hamiltonian, and particle number and composition are conserved. Eq. 1.3 shows the characteristic equation for NVE.[19]

$$m_i \frac{d^2 \mathbf{r}_i}{dt^2} = -\frac{\partial U(\mathbf{r})}{\partial \mathbf{r}_i} \tag{1.3}$$

where m_i is mass of particle i, $\mathbf{r}_i$ is particle i's position in Cartesian coordinates (x_i, y_i, z_i), and t is time. The common numerical solution used is Velocity-Verlet algorithm, shown in Eq. 1.4.[20] The algorithm conserves the system's total energy.

$$\begin{aligned} \mathbf{r}_i(t + \delta t) &= \mathbf{r}_i(t) + \delta t \cdot \mathbf{v}_i(t) + \frac{\delta t^2}{2} \frac{\mathbf{f}_i(t)}{m_i} \\ \mathbf{f}_i(t + \delta t) &= \mathbf{f}_i(\mathbf{r}_i(t + \delta t)) \\ \mathbf{v}_i(t + \delta t) &= \mathbf{v}_i(t) + \frac{\delta t}{2} \frac{(\mathbf{f}_i(t) + \mathbf{f}_i(t + \delta t))}{m_i} \end{aligned} \tag{1.4}$$

where δt represents timestep, $\mathbf{v}_i$ and $\mathbf{f}_i$ represent velocity of particle i and the force acting on it, respectively. The second step in Eq. 1.4 is a force evaluation using the potential.

The canonical NVT ensemble is meant to replicate experimental condition with fixed number of particles, system volume and temperature. The equations

of motion are usually modified, and a common thermostat is Nose-Hoover chains.[21]

Similarly, the isothermal-isobaric NPT ensemble replicates experimental condition with fixed number of particles, system pressure and temperature. Again the equations of motion are modified, and Nose-Hoover thermostat, barostat chains are commonly used.[22,23] Tuckerman et al. showed that the above-mentioned NVT and NPT algorithms replicate their respective ensemble probability distribution.[18]

MD in practice

In running an MD simulation, it is advised to decide the purpose of the simulation, which will determine the most suitable ensemble. As an example, diffusion calculation is most suited to NVE or NVT ensemble. Next, the researcher may decide the initial atomic configuration, including number and composition of particles, and boundary conditions. Periodic boundary conditions indicate that the image is repeated on a Cartesian coordinate, and particles can move across a boundary and re-enter from the other side. Non-periodic boundary is also an option. Some properties may be affected by number of particles simulated (finite-size effect). Empirical adjustment from the literature can be used if available, otherwise multiple simulations at different sizes can be run. The initial conditions such as temperature and pressure may also be chosen, even for an NVE simulation. The desired temperature can be used to initialize particle velocities, and the atomic density can be adjusted to reach the target pressure. Initial configuration may be read from a data file, or generated in the MD software and minimized or equilibrated. A particular potential as described in Section 1.2.1 also needs to be selected.

Experimental runs are likely necessary to adjust the timestep, thermostat, and barostat variables if applicable. The timestep cannot be too large or the simulation will become unstable, and for metal systems timestep of 0.1 to 1.0 fs is typically used. The temperature and pressure should fluctuate around the set temperature and pressure, and thermostat, barostat variables should be adjusted so that fluctuations are within an acceptable tolerance.

Short simulations mimicking the planned production simulation are advised. The researcher can decide which simulation data to collect and at what frequency. Usually thermodynamic data and particle positions will be collected at some frequency, and other types of data to collect depend on the calculations to be performed. For example, mean squared displacement may be collected for a diffusion simulation. The frequency of data collection should be balanced with available memory, especially for particle properties. It is recommended to visualize the thermodynamic quantities throughout the course of the simulation to check for normal, expected behavior. The researcher can also visualize the atomic configurations along the trajectory to behave as expected, using software such as `Ovito`.[24]

After initial tests, the production MD run can be set up. The simulation usually has a minimization, equilibration, and data-collection phase. Equilibration time depends on the initial configuration and the target system state. After equilibration, the thermodynamic properties should be stable and fluctuating around a target.

After the simulation and data collection, the researcher can validate that the entire trajectory behaves according to the target system. The desired properties can be calculated and compared with experimental data or other simulations in literature. Sometimes additional properties may be calculated and used as validation such as radial distribution functions and coordination

numbers for liquids. Repeated production simulations with different initial atomic configurations and velocities are suggested to report variance of calculated properties.

1.2.3 Machine learning

Machine learning is the study of algorithms that improve with experience, measured by their performance on some tasks.[25] Experience refers to data; examples of tasks are classification, regression, clustering; and examples of performance measures are accuracy for classification and mean squared error for regression. Some of the main approaches are supervised and unsupervised learning. In supervised learning, the data is labeled with desired outputs, and the goal is to learn a model from inputs to outputs. Some tasks in supervised learning are classification and regression. In unsupervised learning, data is unlabeled, and the algorithm finds structure within data on its own. Some tasks in unsupervised learning are clustering and dimension reduction.

An ML model is defined by its mathematical definition and training dataset. In complex models, the mathematical definition may be best defined with a computing program (code) and includes the model structure, parameters, and hyperparameters. The training data is most likely preprocessed and modified to normalize and standardize them. The ML models learn from the training data, for supervised learning usually minimizing an error function on the training data, and for unsupervised learning running its algorithm on the training data. In parameterized models, the training process determines the model parameters. We take the model parameters, the error function, exact training dataset with its modifications, and all aspects of the model's mathematical definition to be the ML model.

After a model is trained, we validate it by testing performance on held-out datasets, usually called test or validation sets. If the model performs well on training data but poorly on held-out data, the model has overfit to training data. We typically modify or iteratively retrain models with additional data to improve performance. Then the model can be used in production and predict on new data. We want the model to perform well on new data, i.e. generalize. In real world applications, a common scenario is that the new data is very different from training data (dataset shift or extrapolation), and the performance is not good in these cases.[26] This motivates uncertainty quantification for ML models; also we would continuously update the model over time.

The models used in this dissertation include neural networks (Chapters 2, 3, 4) and Gaussian process (Chapter 5). We used different methods including Bayesian regression (Chapter 5), agglomerative clustering (Chapter 3), nonlinear dimension reduction (Chapter 2). We briefly describe neural networks here and the remaining models and methods in their respective chapters.

Neural network (NN) is a type of model with a set of computations performed in order, with "layers" of matrix multiplications followed by nonlinear activation. Fig. 1.2 shows a representation of a basic neural network with input layer, one hidden layer, and output layer. The circles represent dimensionality of the input and output layers, and number of nodes of the hidden layer.

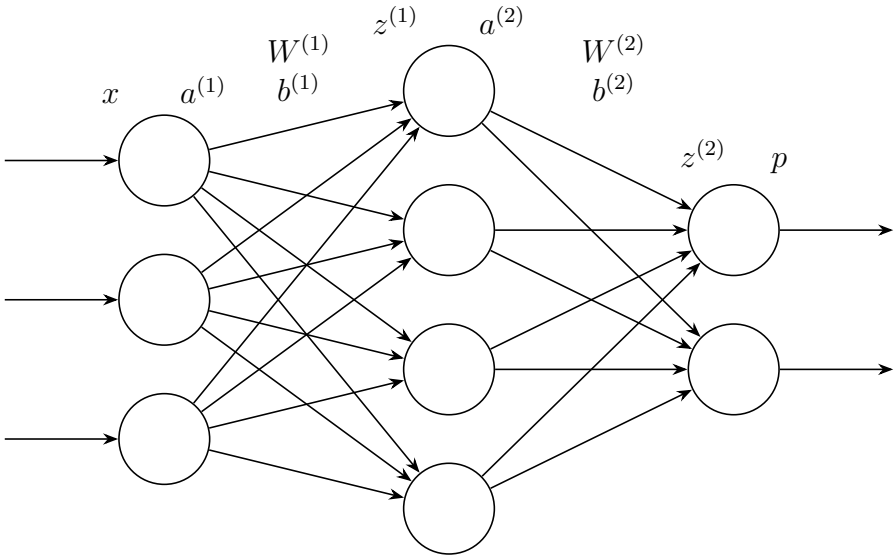


Figure 1.2: Single hidden layer, fully connected neural network representation.

For an input matrix x with n data points and d features, $x \in \mathbb{R}^{n \times d}$, we compute the NN output as:

$$\begin{aligned} &\text{set } x = a^{(1)} \\ &\text{for } i = 1, ..., n_H : \\ &\quad z^{(i)} = a^{(i)} W^{(i)} + b^{(i)} \\ &\quad a^{(i)} = \phi(z^{(i)}) \\ &z^{(n_H+1)} = a^{(n_H+1)} W^{(n_H+1)} + b^{(n_H+1)} \\ &p = \psi(z^{(n_H+1)}) \end{aligned} \tag{1.5}$$

where n_H is the number of hidden layers, ϕ and ψ are activations, and W and b are weights and biases. W have dimension "previous layer dimension" $\times$ "current layer dimension". In the Fig. 1.2 example, $W^{(1)} \in \mathbb{R}^{3 \times 4}$ and $W^{(2)} \in \mathbb{R}^{4 \times 2}$. Biases b have "current layer dimension"; in the example, $b^{(1)} \in \mathbb{R}^{1 \times 4}$, $b^{(2)} \in \mathbb{R}^{1 \times 2}$, also $p \in \mathbb{R}^{n \times 2}$. The activations ϕ and ψ are applied element-wise to each element of z. Sigmoid, hyperbolic tangent, rectified linear unit (ReLU)

are commonly used functions for ϕ, and identity and softmax are usually used for ψ in regression and classification, respectively.

The parameters of the weights and biases matrices are found by minimizing least squares (regression) or cross-entropy loss (classification) between NN prediction and data labels. Stochastic gradient descent is commonly used as the optimization method. For gradient descent methods, given the loss function L, we need to find $\frac{\partial L}{\partial b^{(i)}}$ and $\frac{\partial L}{\partial W^{(i)}}$. These gradients are found using backpropagation, a method to efficiently calculate gradients by using the chain rule, working backwards from the output, and caching relevant quantities. For example $\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial W^{(2)}}$ and $\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial b^{(2)}}$ share the terms $\frac{\partial L}{\partial p}\frac{\partial p}{\partial z^{(2)}}$. The earlier layer derivatives $\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(1)}}\frac{\partial z^{(1)}}{\partial W^{(1)}}$ and $\frac{\partial L}{\partial b^{(1)}}$ also have shared terms with the layers following them, which motivates working backwards through the NN. Writing the derivatives this way is much easier than taking derivatives of the NN written as a forward function composition.

Automatic differentiation (AD) in ML software stores the forward computations as a graph and automatically computes the requested gradients using backpropragation. We used AD for convenient training of NNs (example in Chapter 4), calculation of Hessian and gradients (Chapter 4), Gaussian process derivatives (Chapter 5), and gradient of evidence lower bound w.r.t. variational distribution parameters in variational inference (Chapter 5).

One of the main motivations for ML in this dissertation is ML potentials, which can be orders of magnitude faster than DFT at high accuracy. We trained NN models on DFT data and integrated them in MD simulation (Chapter 2). NN and other ML models are also useful as function approximators, especially when the true function form is unknown. The

universal approximation theorem states that a single hidden layer NN with nonconstant, bounded activation function can approximate any continuous function arbitrarily well with enough nodes.[27] We used NN and GP as function approximators in Chapters 3 and 5. ML methods such as clustering (Chapter 3) and dimension reduction (Chapter 2) may also be useful to find structure of data, and other methods such as Bayesian regression and GPs can be used to find uncertainty of physical properties (Chapter 5).

Uncertainty

Regarding uncertainty, some ML models are known to extrapolate poorly in regions of low or missing data. It is nonobvious how to define an extrapolation region if the ML model has a high dimensional input space. Motivations for uncertainty measurement are to indicate confidence in model prediction and help determine when the model is extrapolating, or is simply not reliable. We can quantify uncertainty using standard error and confidence interval or finding probability distribution of the quantity of interest. In Chapter 4, we find standard error of model prediction using the delta method for parameterized models trained by nonlinear regression, and show examples on atomic systems and NN potentials. In Chapter 5, we compare uncertainties from delta method, Bayesian nonlinear regression, and Gaussian process for physical properties derived from equation of state.

1.3 Organization of dissertation

Chapter 2 describes a motivation of modeling molten superalloys used for turbine blades. Previous work used *ab initio* MD (AIMD) to calculate alloy densities at extremely high computational cost. We conducted extensive hyperparameter tests and trained Behler-Parrinello NN potentials to model the system. We iteratively retrained on additional diverse data and

integrated the NN with custom MD code. The NN potential gave good results compared with AIMD in NVT simulation, and we discussed the challenges faced with the NN potential.

Chapter 3 describes liquid Al-Si system which was studied through experiment and MD simulation. Experiments found significant hysteresis and unique microstructures which motivated MD simulations to investigate the atomic origin of the behaviors, with conflicting results in the literature. We performed MD simulations with a physical potential at various liquid temperatures, and found a Stokes-Einstein deviation near the melting point. We analyzed icosahedral clusters and found that they can explain the observed Stokes-Einstein deviation.

Chapter 4 describes the delta method for uncertainty quantification. In contrast to other common methods, the delta method uncertainties can be found for general classes of models, and completely after a model has been trained. It works well for models with around 1,000 parameters or less, and approximations can be made for larger models. We applied the delta method on simple examples and a NN potential, and show that high uncertainty from delta method correlates with extrapolative input.

Chapter 5 describes different sources of uncertainty in modeling experimental data and presents an analysis comparing methods for the uncertainties they capture. We examine physical properties derived from bulk equation of state model, and three methods of modeling and their uncertainties. To find uncertainties with quantities involving function second derivatives, we develop a Gaussian process with joint over function, first and second derivatives. We show that the uncertainties from our nonlinear and Bayesian regressions are parameter uncertainties while Gaussian process variance includes model selection uncertainty.

In Chapter 6, we conclude and discuss future directions.

2 Machine Learned Potentials for Complex Alloy Systems

2.1 Introduction

There is an increasing demand for energy and efficient power generation. To improve efficiency, turbines and engines are operated at higher temperatures and pressures. A challenge is finding materials, for the turbine parts, that remain chemically and mechanically stable at the conditions. Currently, superalloys of Ni, Al, W, Re and other elements are used.

Superalloys make up turbine engine blades and jet engines. Operating conditions reach temperatures over 1000 °C and pressures over 5000 psi, for high efficiency. The Ni based superalloys are one of few materials that remain structurally stable at these conditions. However, the manufacturing process of the turbine blades may introduce defects. Manufacturing the blades is expensive and involves pouring molten alloy into a mold and cooling it. The cooling process is difficult to experimentally probe and monitor, and thermodynamic data is limited. In addition, the airfoil geometry and compositional chemistry are becoming more complex, increasing the risk of defect formation. Therefore, quantitative models are needed for understanding the properties of this system.[28–32] An accurate model can be used to optimize the alloy processing and predict conditions leading to defect formation.

In previous work, the superalloy model used molecular dynamics (MD) simulation based on density functional theory (DFT). The model accurately predicted molar volumes and diffusion data, compared with available data.[28] However, DFT calculations made the simulations very slow. One nanosecond

of cooling a few hundred atoms required one million CPU hours, equivalent to 1,000 single core computers working for 42 days. The computational time limits the number and complexity of the simulations. For example, it is desirable to simulate larger systems to increase accuracy with respect to the trace elements that are present.[28] It is clear that a faster calculation is necessary to more effectively model the system.

2.1.1 Machine learned potential

An atomistic potential to calculate total energy and forces is necessary for the MD simulation. DFT is one option, which is accurate but computationally slow. Classical, physics-based potentials, such as Lennard-Jones or ReaxFF, have also been used. They are fast but may lack the required accuracy, and are not systematically improvable.[17] Machine learned (ML) potentials are fast, flexible mathematical models which approximate DFT calculations at high accuracy.[33,34] Previous work with ML models is promising; they were successfully incorporated into MD simulations at the necessary level of accuracy, while speeding up computational time by four orders of magnitude.[35–38]

An ML potential requires a mathematical form and a representation of the configuration of atoms, called a fingerprint. The ML potential also requires training data, which is used to fit its parameters by minimizing an error metric, a process called training. The mathematical form, fingerprint, and data are selected so that the ML potential is accurate for the atomic configurations present in the simulation. The next sections discuss background on ML potentials and considerations when selecting the model, fingerprints, and data.

2.1.2 ML potential form and fingerprints

There are many possible mathematical forms and fingerprints for ML potentials. One commonly used method is neural network (NN) with Gaussian fingerprint proposed by Behler and Parrinello (BP).[39] Examples of other methods are Gaussian process regression with "smooth overlap of atomic positions" (SOAP) kernel presented by Bartok et al.,[40] linear regression,[41] kernel ridge regression,[42] and k-nearest neighbors.[43] In addition, there are many proposed fingerprints including Zernike, Coulomb matrix, bag of bonds.[43]

Fingerprints are used because they have desirable properties for ML models as opposed to Cartesian coordinates (properties such as invariance to translation, rotation, and atomic permutation). Fingerprints can be atom-centered (local) or global (describing the entire configuration).

The ML model and fingerprint should be selected such that the ML potential is accurate over the variety of atomic structures appearing in the simulation. This work focuses on the BP NN potentials. BP NN potentials use local energy contributions from each atom and a separate NN for each chemical element. The NNs output atomic energies, and summing over them gives configuration energy. The flexibility of NNs allows the potential to be accurate for the diversity of structures appearing in the superalloy simulation. Another benefit of BP NN potentials is that the model complexity can be easily adjusted, by changing the structure of the NN, and consequently the number of parameters.

This work uses G^2 and G^3 fingerprints.[44] G^2 fingerprints are essentially a coordination number weighted with exponential decay, and G^3 fingerprints capture information about triplet configurations of atoms and inter-atomic bond angle. The dimensionality of the fingerprint needs to adequately

differentiate structures in the simulation, but using fingerprints with higher dimensionality requires more training data and time. As an example, Botu et al. found that local fingerprint vector size of eight worked well for the application of bulk and surface Al.[42]

2.1.3 Training data for ML potentials

In the course of an MD simulation, there will be structures which were not present in the ML potential's training dataset. To be useful, the ML potential needs to be generalizable, i.e. to accurately predict energies and forces for the new structures. Therefore, it is critical that the training dataset includes atomic structures that are representative of those in the MD simulation.[26,45] In addition, generating training data with DFT could be computationally expensive, so it is important to carefully consider the type and amount of training data necessary. This section will describe current methods of selecting training data and interpreting similarity between atomic configurations.

Selecting training data

Many potentials are trained using MD simulation data.[42,46] Often, the type and number of training data for potentials is chosen based on empirical experience and observed success in predictions.[47] There are automatic data generation methods proposed using random structure searching,[48] evolutionary sampling,[26] and active learning.[49] The active learning method proposed by Smith et al. decreased the necessary training data by 90% and decreased the potential's error in an example with organic molecules.[49] This work uses data from MD simulations, but the other data generation methods could be considered in future work.

Interpreting similarity between configurations

Defining similarity between configurations in the simulation and the training set is nonobvious, because there are many degrees of freedom and therefore high dimensionality. This is an active area of research with many different proposed measures of similarity between atomic environments.[50–53]

Dimension reduction often assists with clustering similar data and interpreting the input space. Cubuk et al. use a dimension reduction method (t-distributed stochastic neighbor embedding (t-SNE))[54] on layers of a neural network, and the transformation of data in some layers was able to cluster different phases of Si.[55] Another dimension reduction technique is sketch-map, specifically designed to overcome uneven distribution of states from molecular dynamics trajectories.[56,57] This work uses some dimension reduction techniques to interpret similarity between the data.

Data sampling schemes

Previous experiments have shown that different training data sampling schemes lead to varying errors in ML potentials. Lorenz et al. tested varying density meshes (dense grid, high-symmetric configurations, enhanced lateral grid) from a potential energy surface (PES). The training data scheme varied the NN potentials' error from low error to unacceptably high error.[58] Other experiments tested random vs. farthest point sampling (FPS),[59] and random vs. force binning vs. clustering.[45] They found that random sampling generally under performs. These examples show that the number and configuration of training data make a significant difference in ML potential accuracy. This work also tests sampling methods and relates them to the resulting accuracy and uncertainty of the ML potential.

2.1.4 Objectives

The objective of this work is to create a NN potential which accurately predicts energies and forces for molten Ni-Al-W. We determine efficient and accurate hyperparameters and training procedures, and integrate the potential in MD simulation. To ensure the model does not extrapolate during the simulation, we need to fit the model with enough configurations to cover the potential energy surface. We show how quantitative uncertainty and data visualization aid in model development.

2.2 Methods

2.2.1 Data

The data used in this work are MD trajectories generated using DFT calculations and VASP.[60] DFT calculations were run by Prof. Widom and Dr. Bojun Feng. The DFT data were generated from temperatures varying from 1720-4000 K, different volumes, number of atoms from 32-500, and some varied compositions around the proportions $Ni_{25}Al_5W_2$. Table 2.1 shows the details of the datasets, and additional details about the source of the data is in Appendix A. Fig. 2.1 shows example configurations. Datasets E and G were generated through Monte Carlo MD (MCMD).

Table 2.1: Dataset descriptions. No. atoms indicates number of atoms per structure. Stress column indicates if stress data is available in dataset.

	No. Atoms	No. Structures	Composition	Stress	Description
A.	32	13,000	$Al_5Ni_{25}W_2$	No	T=4000 K; high, low, medium volumes
B.	500	400	$Al_{70}Ni_{415}W_{15}$	Yes	subgroups: mcmd_2080, mcmd_2710, swf0_1, swf0_7
C.	64	684	$Al_9Ni_{52}W_3$	No	T=2000 K
D.	96	150	$Al_{14}Ni_{77}W_5$	Yes	T=2000 K; high, low, medium volumes
E.	96	3,250	$Al_{14}Ni_{77}W_5$, $Al_{12}Ni_{79}W_5$, $Al_{16}Ni_{77}W_3$, $Al_{17}Ni_{73}W_6$	Yes	T=2200 K; high, low, medium volumes
F.	500	3,000	$Al_{70}Ni_{415}W_{15}$	Yes	
G.	32	15,960	$Al_5Ni_{25}W_2$	Yes	four volumes

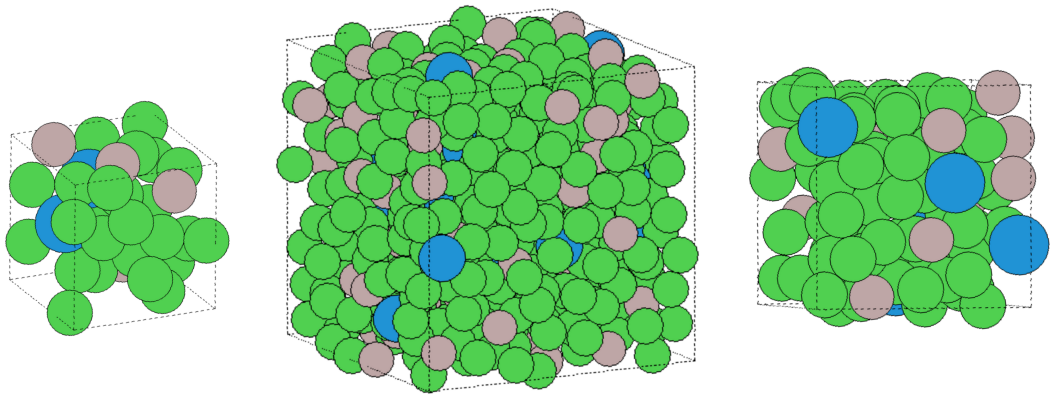


Figure 2.1: Example configurations from Datasets G, B, and D, from left to right.

2.2.2 Software

The ML potentials were trained to minimize both energy and force errors using the Runner software[33,44] and Gilgamesh computing cluster. The MD simulations were custom Fortran codes run by Dr. James Lill.

2.2.3 ML potential hyperparameters

We performed experiments to search for optimal setup of NN structure and symmetry functions. In machine learning terminology, this could be called a "hyperparameter search". In these experiments we used a constant set of training data (subset of Dataset A of Table 2.1), which were 2,080 snapshots of 32 atom cells and constant $Ni_{25}Al_5W_2$ composition. Out of these 2,080 points, 90% were randomly used for each training, and 10% were reserved as the validation set. We used global symmetry functions, meaning the symmetry function setup was repeated for all element combinations. Unless otherwise noted, the NN had (11, 11) nodes with sigmoid activation, symmetry function cutoff radius was 12 Bohr, and results were for 100 epochs of training. In addition, symmetry functions were centered around zero, and energy and force training were used with automatic scaling factor for forces. All energy RMSE are Ha/atom, and all force RMSE are Ha/Bohr.

Eq. 2.1 shows the form of the G^2 symmetry function, Eq. 2.2 shows G^3 symmetry function, and Eq. 2.3 shows the cosine cutoff function. In Eq. 2.2, $2^{1-\zeta}$ is a normalization constant. Fig. 2.2 shows the value of the symmetry function for an atom of R_{ij} distance from the center atom, using a 12 Bohr cosine cutoff. Larger η results in faster decay as expected, and R_s changes the location of the G^2 value peak. Fig. 2.3 shows example angular factors of the G^3 symmetry function, specifically $2^{1-\zeta}[(1 + \lambda \cdot \cos\theta_{ijk})^{\zeta}]$, as a function of the interatomic angle θ_{ijk}. We see that larger ζ contributes to sharper, narrower peaks, and λ changes the location of the peak. For our hyperparameter studies, we checked the number of radial (or pairwise) functions, shifted and centered radial functions, nodes of the NN, cutoff radius, and angular functions. In selecting combinations of G^2 and G^3 symmetry functions, we visualized factors such as in Figs. 2.2 and 2.3 to

decrease overlap or correlation between symmetry functions. We also selected symmetry function parameters to avoid "low variance" symmetry functions, as an example, an all-zero G^2 symmetry function when η is overly large.

$$G_i^2 = \sum_{j=1}^{N_{atom}} e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}) \tag{2.1}$$

$$G_i^3 = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} [(1+\lambda \cdot \cos\theta_{ijk})^{\zeta} \cdot e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \cdot f_c(R_{jk})] \tag{2.2}$$

$$f_c(R_{ij}) = \begin{cases} 0.5 \cdot \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right], & \text{for } R_{ij} \leq R_c \\ 0.0, & \text{for } R_{ij} > R_c \end{cases} \tag{2.3}$$

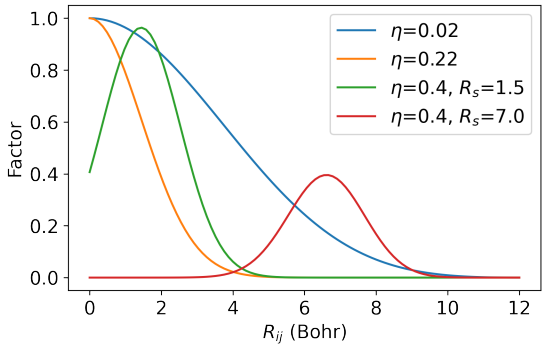


Figure 2.2: Centered and shifted G^2 (radial) symmetry functions.

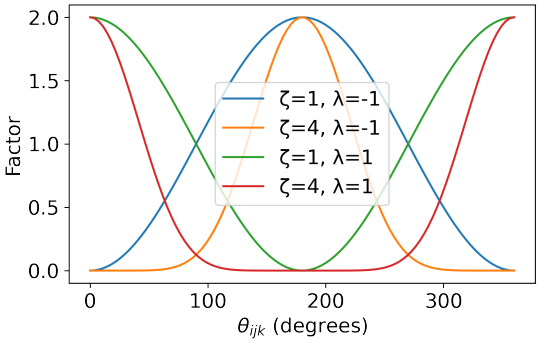


Figure 2.3: G^3 (angular) symmetry functions factors, $\eta = 0$.

2.2.4 Efficient training

To train the ML models efficiently, it is desirable to use less training data to save training time. However, using less training data increases the generalization error. MD trajectories exhibit autocorrelation, and each atom configuration is most similar to its adjacent MD step. An efficient sampling scheme is to sample every n steps, where n can be adjusted to reach the target accuracy on the validation dataset. In a simple experiment, we showed that n-th step sampling is more efficient than random sampling. Using n-th step sampling for four MD runs, the training data has 48 points (12%), while validation has 352 points (88%). The resulting NN potential has a train mean absolute error (MAE) of 0.0008 eV/atom, and validation MAE of 0.0009 eV/atom. In comparison, including the first and last images and randomly sampling 10 points from each MD run (48 total points) results in a train MAE of 0.0007 eV/atom, and a validation MAE of 0.0027 eV/atom. Both potentials have the same fingerprints and (2, 2) NN structure. The n-th step sampling is more efficient because it had lower validation error than the random sampling case. In many machine learning applications, 60-80% of the overall data is used as training data. Using less than 20% of the data as train data is surprisingly efficient and useful for saving train time.

2.3 Results

In this section, we discuss the results from the hyperparameter study. We then discuss the evaluation of the ML potential, including using it to predict on new datasets. To increase diversity of atomic environment in the training data, we iteratively retrain the potential with new data. We discuss the challenges and solutions of integrating the potential with MD simulation and compare the MD simulation results between ML potential and AIMD.

2.3.1 Hyperparameter study

We want to quantify the impact of various parameters on potential performance. We ran experiments testing the radial functions parameters, shifted and non-shifted radial functions, NN architectures, cutoff radius, and angular functions. The exact settings of the models are in Appendix B.

Number of radial functions

For these experiments, we used only non-shifted radial functions (G^2), and varied the number of radial functions. We chose the etas to be spaced evenly across a range of interatomic distance, as an example in Fig. 2.4. Table 2.2 shows the validation energy and force errors and training time. The accuracy improves when radial function sets increase to four. Using three radial functions or more than 10 was less stable for training (sometimes the error would become infinitely large). The error does not improve from 4 to 6 sets of radial functions, but train time increases. Therefore, four sets of radial functions are recommended.

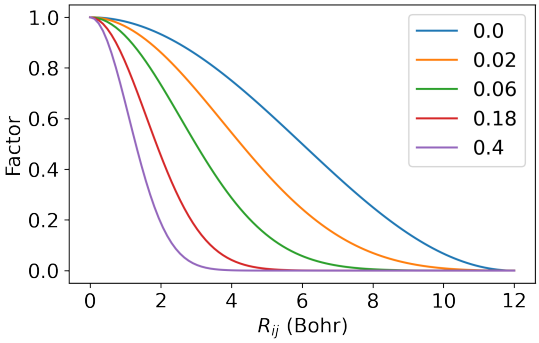


Figure 2.4: G^2 symmetry functions, five sets of etas, evenly spaced across interatomic distance.

Table 2.2: Number of non-shifted radial functions

No. of sets G^2	Val. Energy RMSE (Ha/atom)	Val. Force RMSE (Ha/Bohr)	Time (hrs)
1	0.00109	0.0165	1.0
2	0.00032	0.0049	1.4
3	0.00354	0.1819	1.6
4	0.00019	0.0045	2.1
5	0.00018	0.0044	2.7
6	0.00018	0.0043	7.6
9	0.00022	0.0059	5.0
10	0.01108	0.4438	3.1

Shifted and non-shifted radial functions

In this set of experiments, we tested shifted radial functions, and combinations of shifted and non-shifted radial functions. Each model had four total sets of radial functions. The combinations of functions were selected to try to span the distances between atoms (R_{ij}). Table 2.3 shows the results. The best results with shifted functions reach the same level of error as all non-shifted functions from Table 2.2. From the experiment results, the shifted functions do not make the potential significantly more accurate. Shifted functions are more difficult to select as some of them make

the training unstable. Therefore, non-shifted radial functions are recommended over shifted radial functions.

Table 2.3: Shifted and non-shifted radial functions

No.	Model	Val. Energy RMSE	Val. Force RMSE	Time (hrs)
1	4shift	0.00028	0.0076	3.6
2	1eta-3shift	0.00110	0.0337	5.2
3	2eta-2shift	0.00019	0.0062	2.2
4	2eta-2shift	0.00019	0.0057	2.2
5	3eta-1shift	0.00017	0.0053	2.5
6	3eta-1shift	0.00018	0.0045	2.4
7	4eta	0.00025	0.0045	2.3

Neural net nodes

We tested three different NN structures on the same set of fingerprints. The fingerprints were four sets of non-shifted radial functions with $\eta = 0$, 0.03, 0.08, 0.25. Each NN had two hidden layers with the number of nodes in each hidden layer shown in Table 2.4. When nodes increase from 8 to 11, accuracy increases a small amount, and train time increases a small amount. Increasing nodes from 11 to 15 increases accuracy a small amount, but train time increases significantly. Therefore 11 or 8 nodes is recommended.

Table 2.4: Neural network nodes

NN nodes	Val. Energy RMSE	Val. Force RMSE	Time (hrs)
8	0.00022	0.0042	1.8
11	0.00019	0.0045	2.1
15	0.00014	0.0047	4.2

Cutoff radius

We changed the cutoff radius, while using four sets of non-shifted radial functions. From Table 2.5, when the cutoff radius is decreased to 8 Bohr, the error is still low and training is slightly faster. Increasing cutoff radius to 14

Bohr did not improve accuracy and made training less stable. Therefore, it is recommended to use 8 or 12 Bohr cutoff radius, and 8 Bohr cutoff radius would require more experiments and tests.

Table 2.5: Cutoff radius for symmetry functions

Cutoff radius (Bohr)	Val. Energy RMSE	Val. Force RMSE	Time (hrs)
8	0.00024	0.0042	1.8
9	0.00030	0.0061	1.9
12	0.00019	0.0045	2.1
14	0.00206	0.0594	2.4

Angular functions

We tested various combinations of angular functions, and Table 2.6 shows the successfully trained models that had angular functions and four radial functions. For all of the angular functions used, λ was 1 or -1, ζ varied from 1 to 24, and η was 0. With one angular function, the best performing angular function (used by Model 1 in Table 2.6) is shown in Fig. 2.5. With two angular functions, the best performing angular functions (from Model 6 in Table 2.6) are shown in Fig. 2.6. We found that angular functions with sharper peaks near 180° did not improve accuracy compared with the angular function in Fig. 2.5, meaning ζ values > 1 when $\lambda = -1$ were not as helpful. Also, picking up variation in angles around 0° was helpful for decreasing error, as shown in Fig. 2.6. Therefore when $\lambda = 1$, angular functions with large values of ζ were more helpful. Model 6 of Table 2.6 appeared to have decreasing error after 123 epochs, so we trained it for an additional 100 epochs, which required another 8.6 hours and reached validation energy RMSE 0.00009 and force RMSE 0.0047. This was the lowest energy error among all of the models, but there may be other models that reach similar accuracy with faster train time. Generally, angular functions are more difficult to select, make training

less stable, and take longer to train. For these reasons, angular functions are not recommended unless it is clear that a specific angular function would significantly improve the ML potential.

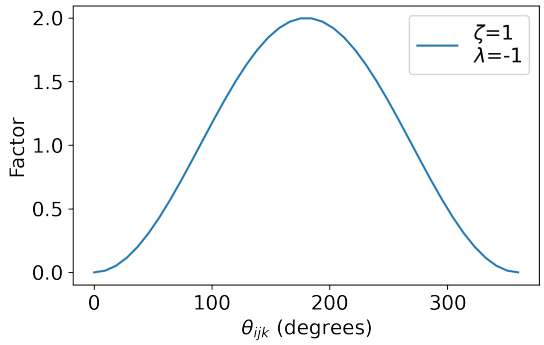


Figure 2.5: Best one angular function model.

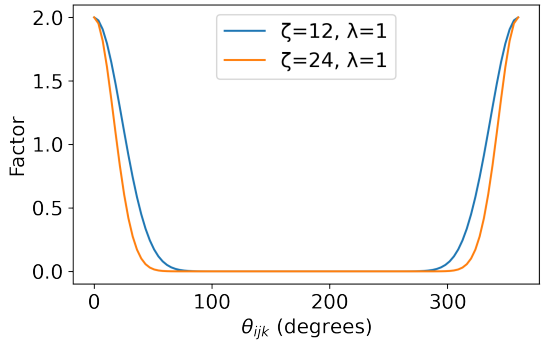


Figure 2.6: Best two angular function model.

Table 2.6: Angular functions and four radial functions

No.	Model	Val. Energy RMSE	Val. Force RMSE	Time (hrs)	Epochs
1	1ang	0.00019	0.0062	5.1	100
2	1ang	0.01229	0.5221	4.6	100
3	1ang	0.00057	0.0190	5.6	100
4	1ang	0.00027	0.0089	5.3	100
5	2ang	0.00098	0.0321	8.3	105
6	2ang	0.00012	0.0049	8.5	123
7	2ang	0.00019	0.0111	8.9	100
8	4ang	0.00116	0.0398	15.0	112
9	4ang	0.00068	0.0234	15.1	100

Recommendations from hyperparameter study

The results from the hyperparameter study quantified the effect of different radial functions parameters, shifted and non-shifted radial functions, NN architectures, cutoff radius, and angular functions. The symmetry functions were selected manually with aid from visualization, and visualizing them is recommended. Increasing the NN nodes, number of radial functions, and including angular functions decreased error, while also increasing train time. Between alternative models, the errors sometimes differed significantly, even by two orders of magnitude. This indicates that experimentation with the hyperparameters, especially the combination of symmetry functions, is necessary for a low-error potential. The hyperparameter study would likely need to be repeated for a different atomic system, i.e. the best hyperparameters here do not necessarily transfer to a different system.

Parallel random search methods with early stopping could be utilized to automate the hyperparameter search in future work.[61] We noticed that within a few epochs of training, it may be obvious that a particular configuration results in high error and training can be stopped early to start an alternate configuration. Automating the hyperparameter search requires ability to stop and start jobs based on logic, and integration with Runner or another training framework.

2.3.2 ML potential training and evaluation

After training an ML potential, we evaluate its performance. Fig. 2.7 shows the energy and force parity plots for Model 1 of Table 2.6. The train, validation, and test datasets were all from Dataset A of Table 2.1. Train, validation, and test datasets had 1,877, 203, and 10,920 structures, respectively. The energy root mean squared errors (RMSE) were 0.00015,

0.00018, 0.00017 Ha/atom and force RMSEs were 0.0061, 0.0062, 0.0061 Ha/Bohr for train, validation, and test sets, respectively. In particular, the errors are about the same across the three datasets. We would not want to see the test or validation error much higher than train error, a clear sign of overfitting. Fig. 2.8 shows the distribution of errors. For the three datasets, the errors appear centered around zero and normally distributed.

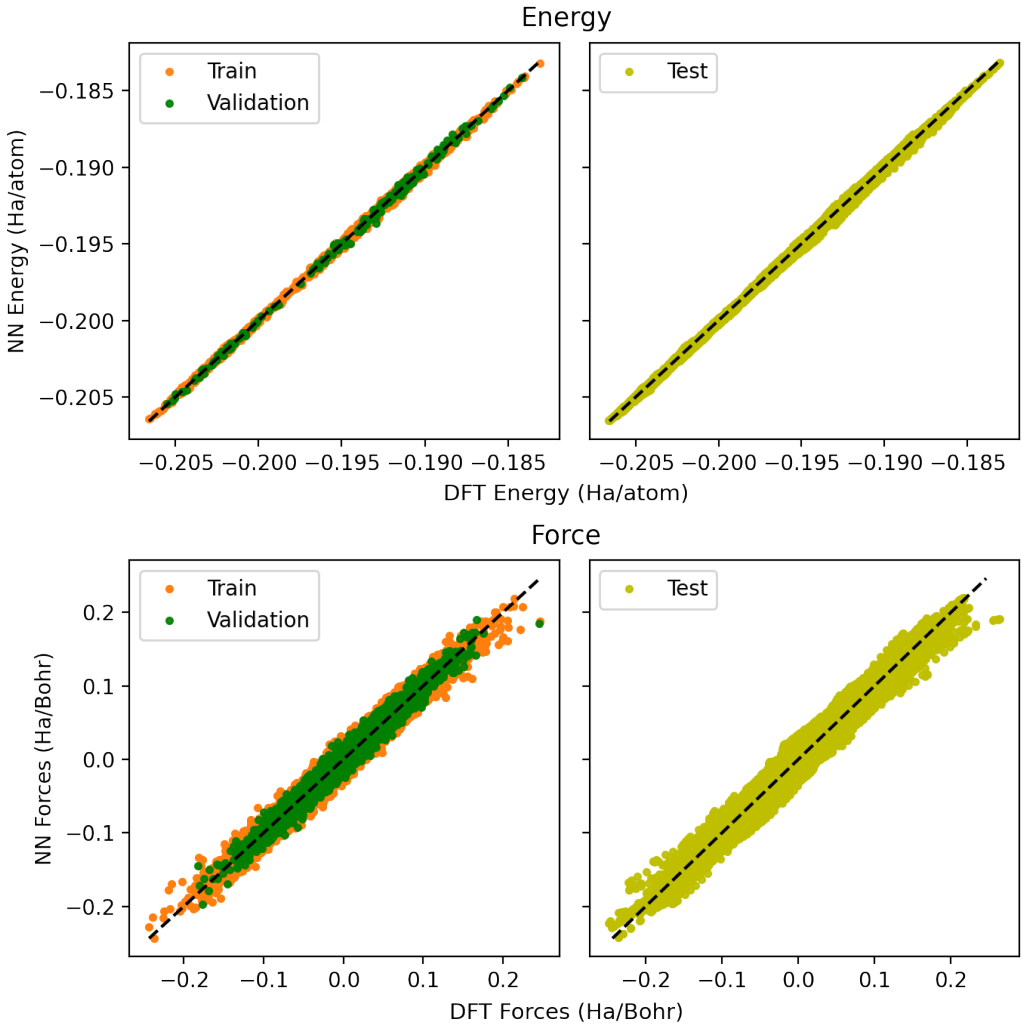


Figure 2.7: Parity plot for energy and forces.

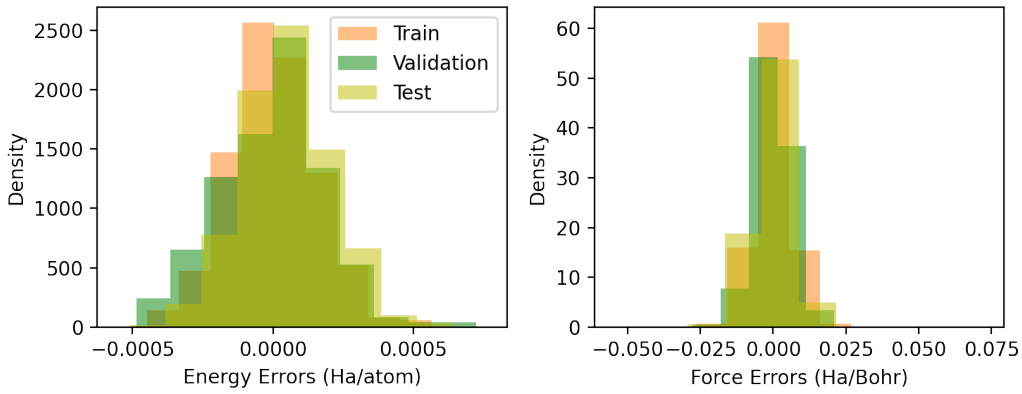


Figure 2.8: Distribution of errors for energy and forces.

Although the ML potential achieved low error for Dataset A, we now evaluate its prediction on a separate dataset. This is more indicative of the ML potential's performance in an MD simulation, since the simulation may encounter structures significantly different than the training data. Fig. 2.9 shows the energy and force parity plots for Dataset B. The energy and force RMSEs are 0.0039 Ha/atom and 0.0066 Ha/Bohr, respectively. In particular, the energy error is much higher than errors for the train, validation, and test sets.

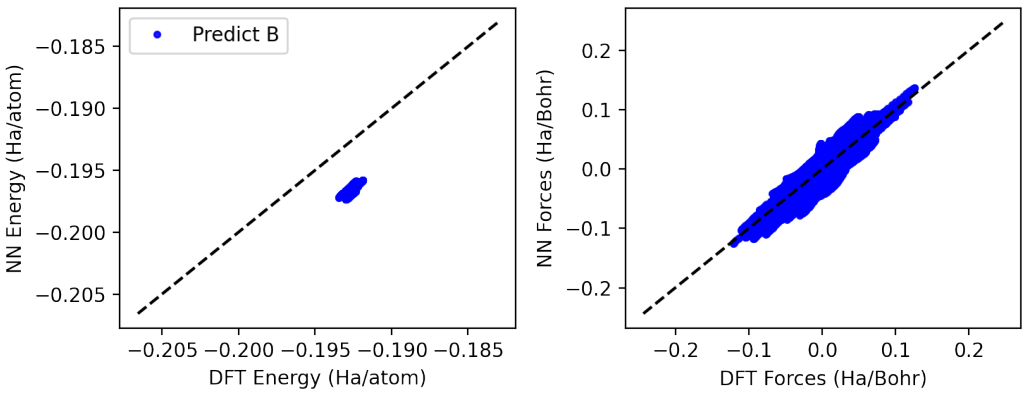


Figure 2.9: Prediction on Dataset B.

The potential seems to be extrapolating on Dataset B. We observe the fingerprint distributions, and Fig. 2.10 shows the distributions of two example

fingerprints for Dataset B and the training data. Dataset B configurations have fingerprints in ranges outside of the training data, which indicates new atomic environments not present in the training data and extrapolation.

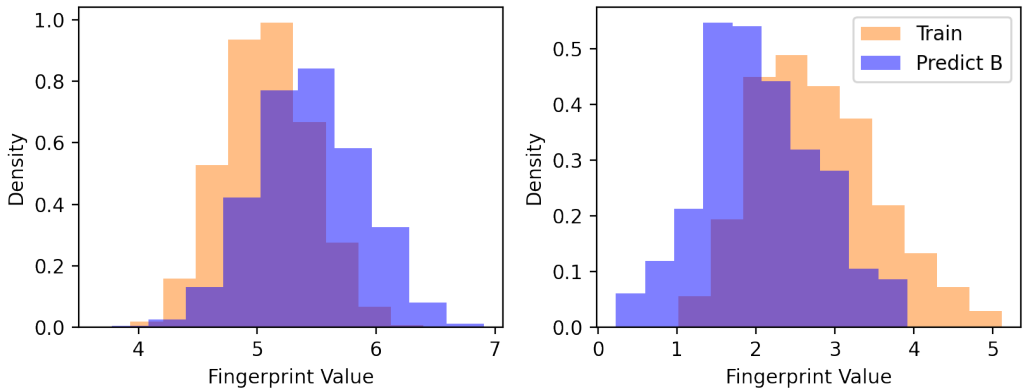


Figure 2.10: Extrapolation on some fingerprints of Dataset B vs. the training set. (Left: Ni center atoms, Ni radial function with $\eta = 0.02$. Right: W center atom, Al-Ni angular function with $\lambda = -1$, $\zeta = 1$, $\eta = 0.0$, cutoff $= 12$.)

We visualize the local atomic environments of 10 MD trajectories, with 100 steps each, from Dataset A. The fingerprint vector of each atom in 1,000 configurations was dimensionally reduced using t-SNE, and Fig. 2.11 shows the result. The "worm"-like structure corresponds with a local environment through one MD trajectory. This indicates that the atomic environment is most similar to its adjacent MD step, and that the finite number of MD trajectories covers an extremely small portion of the true potential energy surface. The visualization further indicates that the potential needs to be trained with more data.

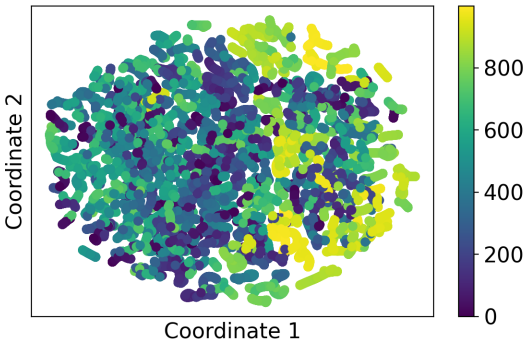


Figure 2.11: t-SNE representation of 10 MD trajectories from Dataset A (atomic environments). Worm-like structures indicate atomic environments are most similar with adjacent MD step. Visual separation between trajectories indicates sparse coverage of true potential energy surface.

To improve the ML potential performance, we added 5% of Dataset B to the training data and retrained. Fig. 2.12 shows the energy and force parity plots after retraining, and Fig. 2.13 shows the distribution of errors. The updated energy and force RMSEs for Dataset B are 0.00015 Ha/atom and 0.0059 Ha/Bohr, respectively. The errors reached the same level as Dataset A's. We showed that adding a small amount of new data and retraining allows the potential to be accurate for the entire dataset. In this sense, the ML model is systematically improvable.

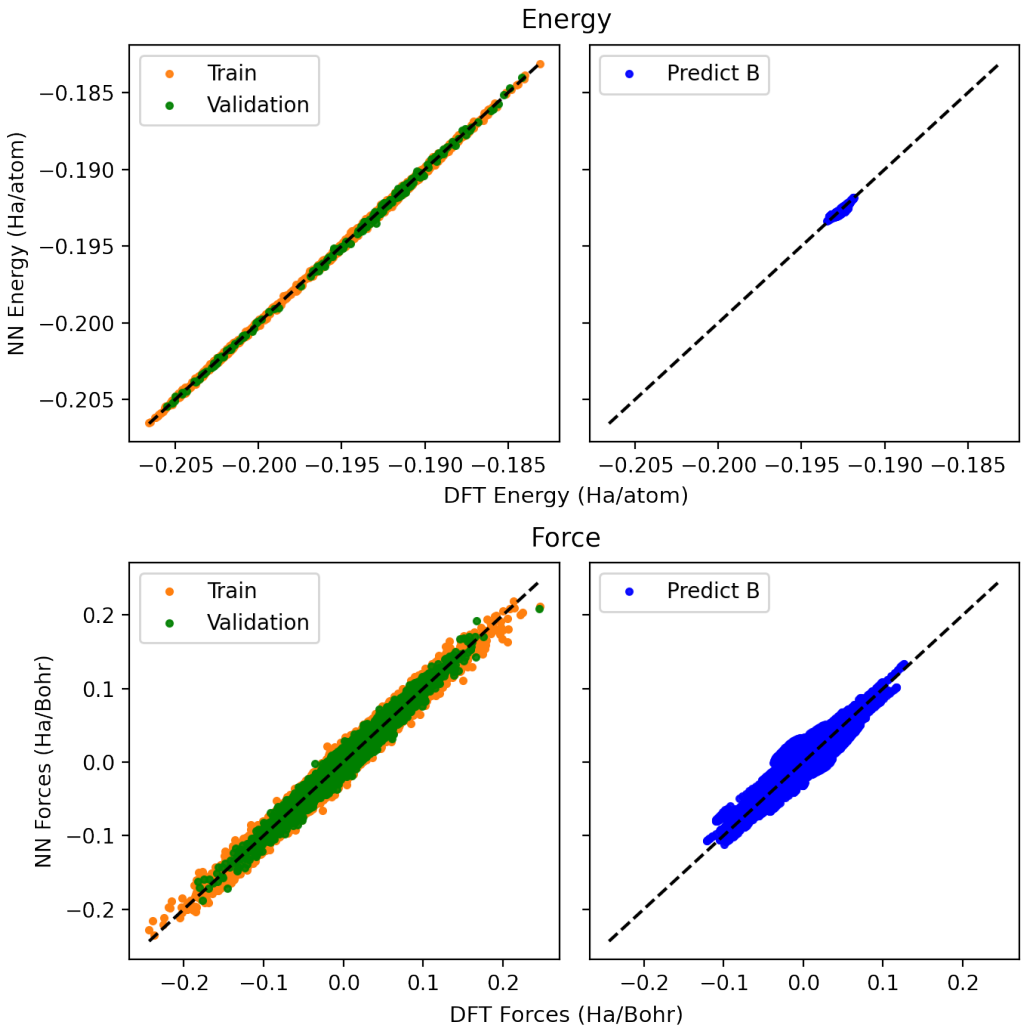


Figure 2.12: Parity plot after retraining on additional 5% of Dataset B.

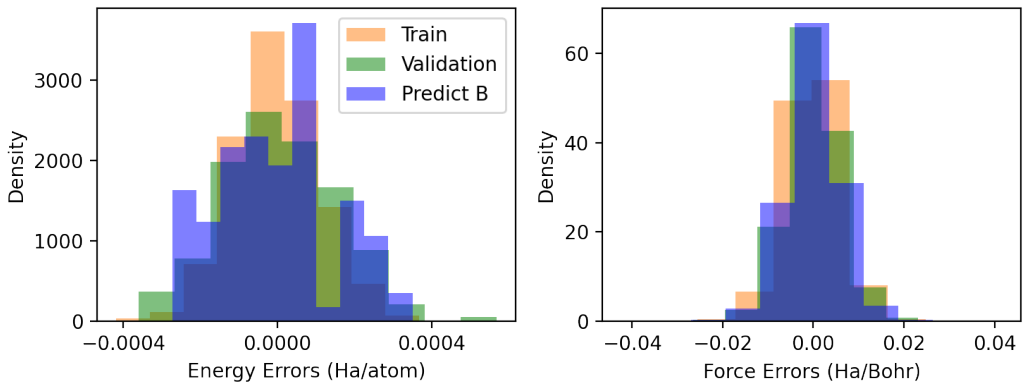


Figure 2.13: Distribution of errors for energy and forces after retraining.

2.3.3 Integration with MD simulation

The process of training, validating the ML model, and adding varied training data was repeated several times. Intermediate models were used in MD simulation and would sometimes extrapolate to unphysical results. In these cases, it was clear that the potential required training on additional data. In effort to increase diversity of atomic environments, we added data with different stresses, volumes, compositions, higher temperatures, and larger number of atoms per structure. Table 2.7 shows the updates made starting from Model 1 of Table 2.6. For each iteration, we started retraining with optimal weights from the previous model. Usually starting with the previous best parameters gave better results than starting with random weights. Table 2.7 also shows other updates to the model, usually to aid integration with the MD simulation. In Model Iteration 4, we changed the force scaling factor to 1.0 to decrease force error, which had the intended effect. In Model Iteration 6, we removed the W-W angular function because it was zero for all atoms. This is because W is sparse in chemical composition and a triplet of W atoms is highly improbable. We found that Runner always scales symmetry functions for its stress prediction, and therefore including the W-W angular function resulted in infinite stress prediction, which negatively impacted the MD simulation results. In Model Iteration 6, we also scaled symmetry functions to improve the stress prediction, as scaling is consistent with Runner's stress calculation. This change reduced the stress RMSE from 0.025 to 4.8e-5 Ha/Bohr^3, which greatly improved the stability and reasonableness of the MD simulation. Fig. 2.14 shows the stress parity of 500 random test structures (100 each from Datasets with stress) for Model Iteration 6. In future work, stress training could be included to further decrease the stress error.

Table 2.7: Iterative retraining of model

Iteration	Training Data (No. Structures)	Updates
1	Dataset A (1,877)	
2	+ Dataset B (16)	
3	+ Dataset C (33) + Dataset D (14)	
4	+ Dataset E (152)	Changed force scaling factor from default to 1.0, trained 300 epochs
5	+ Dataset F (259)	Force scaling factor is default
6	Final: Dataset A (1,877), Dataset B (16), Dataset C (18), Dataset D (14), Dataset E (152), Dataset F (259), Dataset G (1,440)	Removed W-W angular function, scale symmetry functions, force scaling factor is 1.0

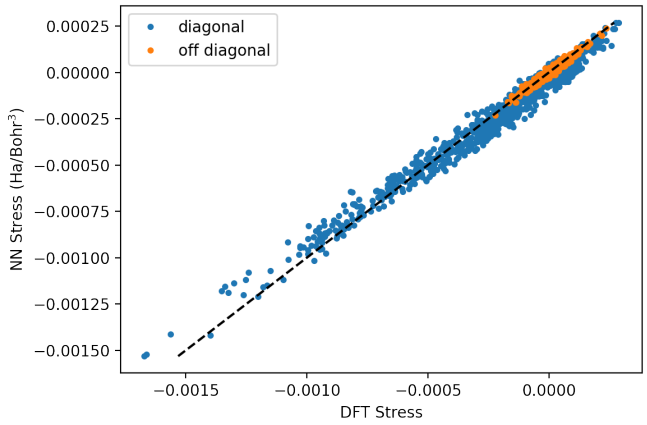


Figure 2.14: Stress parity has RMSE 4.8e-5 Ha/Bohr^3.

The final model of Table 2.7 showed high accuracy on all of the datasets and ran in NVT simulation for over 10^5 fs. The training set contained 3,776 structures. The potential achieved RMSEs of 0.00029 Ha/atom energy and 0.0036 Ha/Bohr forces for both its training and validation sets. The symmetry functions were four sets of non-shifted radial with $\eta = 0.02, 0.05, 0.10, 0.35$, and angular function set with $\lambda = -1$, $\zeta = 1$, $\eta = 0.0$, excluding the W-W-W angular function. Symmetry functions were centered and scaled. The cutoff

radius was 12.0 Bohr. NN had 2 hidden layers with 11 nodes each and sigmoid activation.

We ran a NVT simulation with 500 atoms at 1720 K using Nose-Hoover thermostat. The ML potential raised extrapolation warnings when a fingerprint is outside of its range from the training data. With a stable NVT simulation, we still see around 1,000 extrapolation warnings at each time step. As such, the extrapolation warnings provide limited information in determining the reliability of the ML potential in simulation, and this motivates a need for uncertainty quantification.

We calculated diffusivities using Green-Kubo and Einstein formulas, and compared with AIMD. As an example, Fig. 2.15 shows the Al diffusivities. The uncertainties for the diffusivity estimates are smaller for the ML potential than AIMD. The Green-Kubo and Einstein diffusivities do not exactly match for the ML potential, at maximum differing by 1 cm^2/s, while they do match for AIMD. The diffusivities calculated by these methods should match and indicates the correctness of the simulations.[62] Since they match for AIMD but not the ML potential, it may indicate that the ML potential's fitting errors are not low enough, or some setting in the MD simulation is inconsistent with the ML potential. This can be tested in future work. The ML potential's calculated diffusivities are higher than AIMD's, sometimes by a factor of two, but still the same order of magnitude.

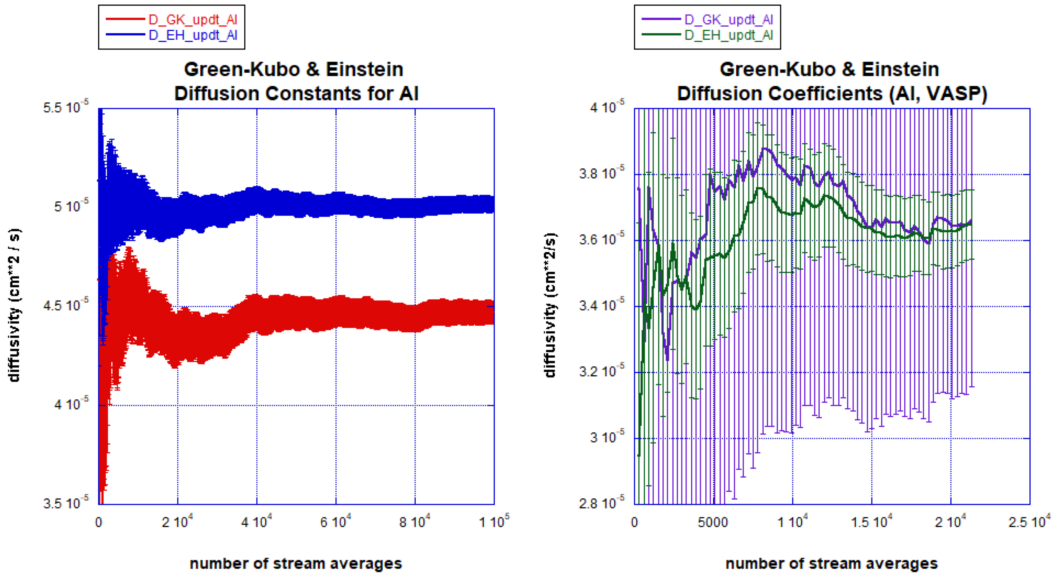


Figure 2.15: Al diffusivity for ML potential (left) and AIMD (right).

We calculated bulk and shear viscosities using Green-Kubo and Einstein-Helfand formulas, and compared them with AIMD. For both viscosities, the ML potential had smaller uncertainties than AIMD, and the ML potential viscosities were smaller, sometimes by a factor of two, but in the same order of magnitude.

We made radial distribution function (RDF) plots for the ML potential and AIMD. The radial distribution functions and location of peaks generally matched between the ML potential and AIMD. There was more scatter on the RDFs from the ML potential and for dilute pairs, and the peak heights were higher. As an example, Fig. 2.16 shows the Ni-W RDF. The atomic volumes and atomic coordination matched between the ML potential and AIMD, with the ML potential having less uncertainty. Additional figures comparing simulation results between ML potential and AIMD are in Appendix C.

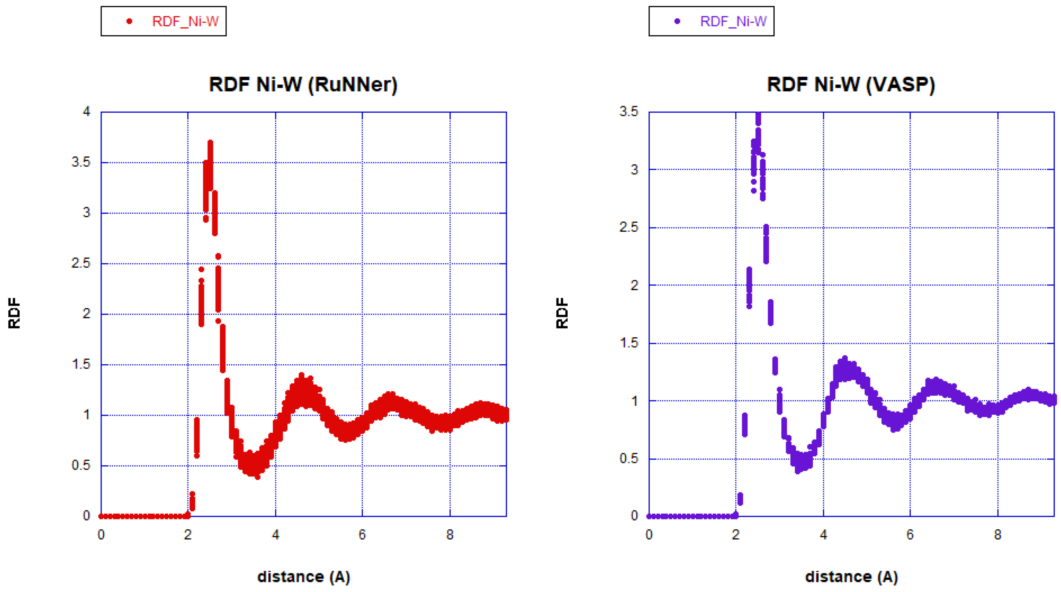


Figure 2.16: Ni-W radial distribution function for ML potential (left) and AIMD (right).

We visualized the final ML potential training data using t-SNE, as shown in Fig. 2.17. Here we included all training data from Datasets A-G except for one-tenth of Dataset F, and visualized the W local environments. We considered W because it is sparse and therefore less covered in the PES. In Fig. 2.17, there are still visible gaps in the 2-D visualization, while there are fewer gaps if we visualize Al or Ni local atomic environments. The gaps could mean that our datasets have not covered the complete PES or that there are regions of fingerprint space that are not energetically favorable. We believe that we still need more data because the current potential became unstable during NPT simulation. This could be tested by continuing to generate data and iteratively training the potential, especially with NPT simulation. We color coded by Dataset A/B/. . . /G. The colors are mostly overlapping, which indicates that this separation among datasets is not necessarily the best way to delineate the fingerprint space. We also tried separating data by volume and temperature but did not notice clear separations in those cases either. Partial

coordination number is another possibility to test delineating the fingerprint space in future work.

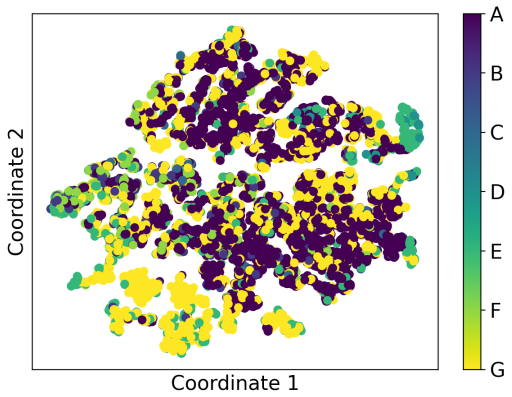


Figure 2.17: t-SNE representation of W local environments in training data of final potential.

As previously mentioned, to run the ML potential with NPT simulation, we would need to increase the training dataset diversity. The current potential was stable in NPT simulation for the first 3,500 fs and then accumulated too many extrapolation warnings and became unstable. The current potential is also reaching a plateau of accuracy. In future work, the potential would probably require more NN nodes. In training the final potential, one update of potential weights required about 50 minutes, which would be even longer if the NN had more nodes. Options that could speed up training are training on GPU or parallelizing the optimization algorithm which would also require changing the algorithm. An example of an easily parallelizable algorithm is stochastic gradient descent. Further research could also determine better ways to select data for training. If we increased the number of species in the liquid alloy simulation, we would likely implement a different style of symmetry functions because the current scheme of symmetry functions increases quadratically with number of atomic species. Currently Runner only implements these BP symmetry functions described.

We found that changing the chemical system changes the optimal symmetry function and NN setup, and we could develop an automated search for the optimal setup.

2.4 Conclusions

In this chapter, we discussed building the ML potential for molten Ni-Al-W. We started with a hyperparameter study, which quantified the impact of various radial and angular symmetry functions, NN nodes, and cutoff radius on the resulting potential's error. Experimenting with different symmetry function setups is recommended for a new system because the resulting potentials can vary greatly in their errors. After training an ML potential, we evaluated its performance on test datasets and more diverse datasets. In the early iterations, the ML potential extrapolated on the new datasets. We added a small amount of the new datasets and retrained the potential, which resulted in low errors for the entire dataset. These findings motivated the process of iterative training as a means to explore the potential energy surface and increase atomic environment diversity in the training data. After several rounds of iterative retraining, we created a potential which worked well in MD simulation for over 10^5 fs. In order to integrate the ML potential with MD simulation, we wrote new custom codes and made modifications to the potential over several rounds of debugging. The simulation results showed good agreement between ML potential and AIMD in terms of radial distribution functions, diffusion constants, and viscosities. During the stable MD simulation, the ML potential still generated extrapolation warnings, indicating a need for a better measure of model reliability or uncertainty. We also found that searching the dataset space was challenging, and knowing what data to select for training and when the model would extrapolate was nonobvious. These factors also

motivate the quantitative uncertainty for ML potentials and similar models. In future work, we need to continue improving the dataset diversity and ML potential accuracy to run stable NPT simulation for this system. The lessons learned about hyperparameter selection, building up the dataset and iterative retraining, and challenges of integrating the ML potential with MD simulation can be applied to future studies in this domain.

3 Origin of the Stokes-Einstein Deviation in Liquid Al-Si

3.1 Introduction

Simulation of materials on multiple length scales is useful to calculate different properties. On the larger continuum length scales, we measure viscosity, density, and may observe hysteresis.[63] At the smallest scale, we simulate atom interactions, using first-principles density functional theory or atomic potentials. One goal is pushing the time and length scale boundaries of atomic simulations and finding relationships between atomic scale and continuum properties.

The Stokes-Einstein equation relates diffusion and viscosity in liquids, an example of relating individual particle properties (diffusion) with the continuum (viscosity). The Stokes-Einstein relation (SER) is shown in Eq. 3.1, and states that the effective diameter d is constant with the ratio $\frac{T}{D\eta}$ where T is temperature, D is diffusion, and η is viscosity. The related fractional Stokes-Einstein (FSE) in Eq. 3.2 is shown to hold for many liquids.[64] Values of the exponent κ usually fall between 0.6 and 1. The FSE with $\kappa = 1$ is equivalent to SER, while other values of κ result in a non-constant effective diameter over temperature, or SER breakdown.

$$d = \frac{k_B T}{2\pi\eta D} \tag{3.1}$$

$$\frac{D}{T} \propto \left(\frac{1}{\eta}\right)^{\kappa} \tag{3.2}$$

There has been much interest in the SER for various liquids including water,[65] liquid metals and alloys,[62,66,67] and micelle system.[68] The SER does

hold for many liquids within a temperature range, however when it holds or breaks down in specific systems is an active area of research. Differing hypotheses for breakdown have been proposed without a consensus reached.[69,70] Some hypotheses are local microviscosity differing from bulk viscosity,[70] increase in local five-fold symmetry and icosahedral clusters,[71] and dynamic decoupling/heterogeneity or large difference in self-diffusion of chemical species.[72,73] Besides understanding liquid particle and continuum properties, the SER has a practical use of estimating diffusion from viscosity or vice versa as they can be difficult to measure experimentally.[74]

The derivation for SER arises from Einstein relation of Brownian motion and Stokes' law for Stokes flow.[75] The assumptions for the Einstein relation are independent particle motion and velocities of particles following Maxwell-Boltzmann distribution. These assumptions result in Brownian motion or random particle motion. The assumption for Stokes flow is spherical particles in low Reynold's number or smooth, non-turbulent flow. From these assumptions, it seems that the SER may break down if particles have strongly directional interactions (non-random motion) or large clusters formed in the liquid (non-spherical particles).

We have an interest in computational models of liquid alloys, as metallurgical processes are moving towards systemization and optimization via simulation. There are many behaviors which are important to understand for alloy processing. These include non-ideality (when entropy of mixing is far from the ideal entropy), hysteresis, local order or short-range order, differences in local densities, and SER holding for liquid but breaking down at low temperatures. Some studies of various alloys found SER breakdown at low temperatures,[69,71,76] and researchers also found onset of clustering/local order.[71,76] Li et al. found that chemical short-range order is not a universal

indicator of SER breakdown, but attributed the SER breakdown to rapid increase in five-fold symmetry and icosahedron cluster formation.[71] Five-fold clusters were found in liquids,[77] and five-fold symmetry indicated dynamic slowdown in liquid glasses.[78] Fig. 3.1 shows an icosahedron cluster, and the five-fold symmetry around the center atom is visible.

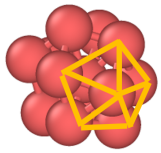


Figure 3.1: Icosahedron cluster (13 atoms) with visible five-fold symmetry.

This work focuses on Al-Si because it is a commonly used alloy in automotive, aerospace, and electronics applications, and demonstrated unusual macro-properties. Studies showed hysteresis, with different small-angle neutron scattering results depending on if liquid was heated or cooled to 950 °C.[63] Al-Si at eutectic composition also forms different liquid structure depending on the thermal treatment.[79–81]

Researchers simulated liquid Al-Si to examine its behavior and possible origins of the hysteresis and microstructures. There have been some *ab initio* MD (AIMD)[79–84] and modified embedded atom method (MEAM)[85] studies calculating coordination numbers, partial correlation functions, diffusion, bond angle distributions, and other properties. The simulations covered mostly eutectic composition (12 at.% Si) and also the entire range of compositions and temperatures from 900-2000 K. Some studies found evidence of chemical short-range order,[79,81,82,84] while others found the liquid was well mixed.[80,85] One study found tetrahedral short-range order[79] while another found negligible tetrahedral structures.[81] There were other interesting findings. For instance, Qin et al. found self-diffusion decoupling of Al and Si in low concentration Si alloys,[82] and Ji et al. found that

coordination numbers and structure factor peak height change faster in 900-1200 K temperature range, possibly suggesting a structural change in the liquid around that temperature.[84] Saidi et al. calculated step mobilities at an Al-Si crystal-liquid interface using MEAM and around 10,000 atoms simulation, but the calculated properties were not all available for comparison.[86] The studies seem to indicate some complex behavior of liquid Al-Si that requires further investigation, and some studies have contradictory results which we would like to resolve. Additionally, previous studies did not examine SER in liquid Al-Si.

Our objective in this work is to investigate the SER and local order in liquid Al-Si using molecular dynamics. Investigating SER requires calculating diffusion and viscosity using data from MD simulations at multiple temperatures including the supercooled liquid range. For the simulations, we use an angular-dependent potential developed by Starikov et al.[14] because it is fast and showed good agreement with Al and Si liquid properties, including diffusion and radial distribution functions. We also test two high Al alloy compositions that are close to the eutectic composition. We focus on local order and icosahedron clusters because previous studies found icosahedron clusters' correlation with SER breakdown[71] and their importance in liquid structure.[77,78,87,88] We analyze the local order using coordination numbers, radial distribution functions (RDFs), Voronoi polyhedrons (VPs), and agglomerative clustering. Finally, we approximate the clusters' effect on viscosity and diffusion using per-atom viscosity and diffusion calculations. The novel contributions of this work are:

1. evidence of Stokes-Einstein deviation in liquid Al-Si near the eutectic composition,

2. the use of agglomerative clustering method applied to analyze atomic clusters,

3. quantifying clusters' effect on viscosity and diffusion using per-atom methods.

Using the novel methods, we show that viscosity increases when icosahedral clusters are present, while diffusion remains the same, which could be the origin of the SER deviation in this system.

3.2 Methods

The MD simulations were done in `LAMMPS` (`http://lammps.sandia.gov`)[89] with periodic boundary conditions, and 1 fs timestep. We ran simulations in the isothermal-isobaric (NPT) and canonical (NVT) ensembles for two compositions $Al_{90}Si_{10}$ and $Al_{95}Si_5$, and we describe their details in the following sections. The purpose of the NPT simulations was to examine the possible phases that could form. We used the NVT simulations to collect data for diffusion and viscosity calculations. We used NPT solid-liquid interface simulations to determine the melting point for the chosen compositions and potential. The calculations were run on Bridges[90] within the XSEDE program.[91] The data and code for this work are in the repository.[92]

3.2.1 NPT cooling and NVT simulations

The simulations for $Al_{90}Si_{10}$ contained 988 Al and 111 Si atoms, and simulations for $Al_{95}Si_5$ contained 1,037 Al and 62 Si atoms, which are 1,099 atoms total for each composition. We generated initial liquid configurations by equilibrating at 1600 K and 0 bar pressure for 2 ns using NPT ensemble. We ran cooling simulations at different cool rates. These simulations started

at 1600 K and cooled to 900 K at three different rates: 1, 0.1, and 0.01 K/ps. The purpose was to examine if differing cool rates result in different phases. Another set of cooling simulations at the three cool rates started at 900 K and cooled to 400 K. All cooling simulations were run in NPT ensemble with 0 bar pressure, temperature ramp, and Nose-Hoover barostat. We also ran constant temperature NVT simulations with Nose-Hoover thermostat for calculating diffusion and viscosity. The simulation box volume was chosen from the cooling simulations to achieve near zero pressures. These simulations were initialized from randomly placed atoms and equilibrating for 20 ps. The data collection simulation was then run for 10-20 ns to decrease uncertainty in viscosity measurement. The simulations were repeated seven times using different atom position and velocity initializations.

3.2.2 NPT melting point simulations

Solid-liquid interface simulations were used to determine melting point. The simulations for $Al_{90}Si_{10}$ contained 1,901 Al and 204 Si atoms, and simulations for $Al_{95}Si_5$ contained 2,020 Al and 121 Si atoms. The FCC-solid and liquid configurations were placed into two equal volume halves of the rectangular simulation box. There were 1,099 atoms placed in the solid region and fewer atoms in the liquid region, corresponding with the liquid density. We used `LAMMPS minimize` to smooth contact at the interface. NPT simulations were run at multiple temperatures with pressure set to 0 bar using Nose-Hoover thermostat and barostat. Simulation length was 500 ps for temperatures near the melting point and 200 ps for temperatures further from the melting point. We plot volume vs. temperature, using the final volumes near the end of the simulations, and take the temperature at which a sudden volume change occurs to be the melting point.

3.2.3 Diffusion

Diffusion was calculated using slope of mean squared displacement, shown in Eq. 3.3,[79] where α represents species and $\langle . \rangle$ represents ensemble average. We used the Einstein method to calculate diffusion because it is robust and less data intensive than the Green-Kubo method.[93] Diffusion follows a finite size effect with diffusion increasing with cell size, therefore we applied the Yeh-Hummer correction shown in Eq. 3.4, where D_∞ is the infinite-cell diffusivity, $D(L)$ is the diffusivity for box length L, k_B is Boltzmann constant, T is absolute temperature, η is viscosity, and $\xi = 2.837$ is a dimensionless constant determined from Ewald-like summation of a periodic lattice.[94] The diffusion coefficients were calculated by averaging the mean squared displacement data from the last 200 ps of the simulation. This ensures the diffusion is in the linear region and stabilized as long-time diffusion.

$$D_\alpha = \lim_{t \to \infty} \frac{\left\langle |R_\alpha(t) - R_\alpha(0)|^2 \right\rangle}{6t} \tag{3.3}$$

$$D_\infty = D(L) + \frac{k_B T \xi}{6\pi\eta L} \tag{3.4}$$

3.2.4 Viscosity

Viscosity was calculated using Green-Kubo integration of stress autocorrelation function (SACF), shown in Eq. 3.5.[95] In our case, any off-diagonal element of stress tensor can be used, and we averaged the three off-diagonal combinations for the final viscosity. The SACF data is collected during the simulation. We collected SACF data for 1-2 ps depending on the temperature and used 10,000 ensemble averages to smooth the SACF. At

some low temperatures, the SACF requires longer than 2 ps to decay to zero but follows a smooth decay path. Therefore, we fit the SACF data using Eq. 3.6 as used by Guo et al.,[96] and $C, \omega, \tau_f, \tau_s, \beta_f$, and β_s are fitting parameters. The parameters were fit using `scipy.optimize.curve_fit`,[97] and Fig. 3.2 shows an example of the SACF data in blue and fit in orange. The SACF is quite smooth but did not reach zero by 2 ps. The fit has some errors, therefore we fit the errors between the SACF function and data using a neural network with one hidden layer, four hidden units, and hyperbolic tangent activation. A larger neural network could be used, but we found that this architecture offered good accuracy and fast optimization. The neural network weights were optimized using `scipy.optimize.minimize` and minimizing the least squared loss. Fig. 3.3 shows an example neural network fit. We expect the residual errors to be near zero as time increases, so it is clear that we should only integrate the neural network residual in the time range where SACF data is available. We integrate using trapezoidal rule: the SACF function from 0-6 ps, and neural network delta in time range where SACF data is available (0-1 or 0-2 ps), and sum the two integrations. Using this two-stage fit method allows us to save computational time by collecting SACF data up to 2 ps and still maintaining accuracy with the SACF function. The neural network fit improves the accuracy while preventing extrapolation. A low variance of the viscosity proves to be essential in the effective diameter calculation.

$$\eta = \frac{V}{k_B T} \int_0^\infty \langle P_{xy}(0) P_{xy}(t) \rangle dt \tag{3.5}$$

$$\frac{\text{SACF}(t)}{\text{SACF}(0)} = (1 - C)\cos(\omega t)\exp(-t/\tau_f)^{\beta_f} + C\exp(-t/\tau_s)^{\beta_s} \tag{3.6}$$

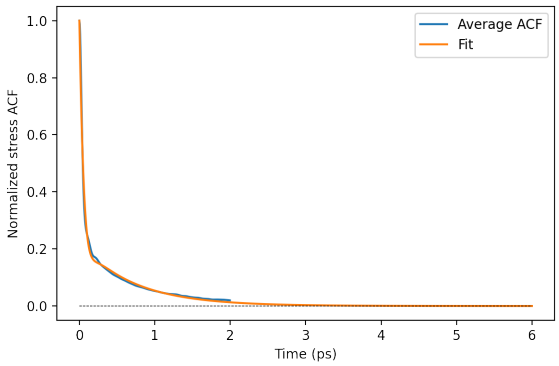


Figure 3.2: Example of stress autocorrelation data and fit.

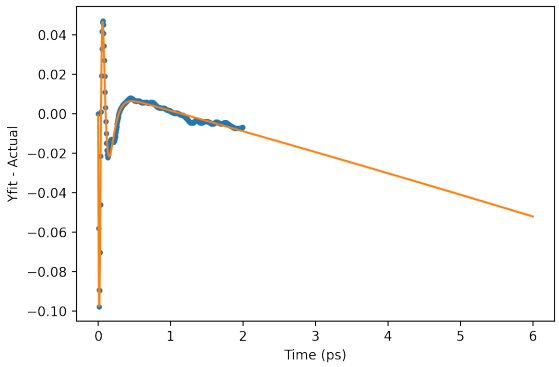


Figure 3.3: Residual fitting error of SACF function minus SACF data using neural network.

3.2.5 Effective diameter

We used propagation of error to find the standard error of effective diameter, which is $\sqrt{(\frac{\partial g}{\partial T})^2 \mathrm{se}(T)^2 + (\frac{\partial g}{\partial D})^2 \mathrm{se}(D)^2 + (\frac{\partial g}{\partial \eta})^2 \mathrm{se}(\eta)^2}$, where $g = \frac{k_B T}{2\pi\eta D}$. We found that the term with $\mathrm{se}(\eta)^2$ contributes the most to the uncertainty, which is why a low variance in the viscosity is required.

We also fitted the FSE relation, and found the exponent κ from the negative slope of the line regressing $\log(D/T)$ vs. $\log \eta$. We report $\pm 2 \cdot$s.e. where s.e. is standard error of the slope, determined from the parameter covariance matrix using `numpy polyfit`.[98] Fitting the FSE exponent allowed us to quantitatively assess if SER deviation occurred.

3.2.6 Radial distribution function

We created radial distribution functions (RDF) using atom position information (`LAMMPS` dump files) of the NVT simulations. We used atom position information every 10 ps, and averaged over 350 snapshots to generate the RDFs. The partial coordination histogram data up to 10 Å cutoff was obtained using `Ovito`,[24] and partial and total RDF are related by Eq. 3.7, where $g(r)$ is total RDF, c_α and c_β are concentrations of the two species α and β, and g_{ij} are the partial RDFs with i being the center atom.

$$g(r) = c_\alpha^2 g_{\alpha\alpha}(r) + 2c_\alpha c_\beta g_{\alpha\beta}(r) + c_\beta^2 g_{\beta\beta}(r) \tag{3.7}$$

3.2.7 Coordination numbers

Coordination numbers were obtained by integrating the partial RDFs as in Eq. 3.8. Here, N_{ij} represents j atoms surrounding an i center atom, R_{min} is first minimum in the total RDF, and ρ_j is number density of j species.

$$N_{ij} = \int_0^{R_{min}} 4\pi r^2 \rho_j g_{ij}(r) dr \tag{3.8}$$

3.2.8 Voronoi tessellation

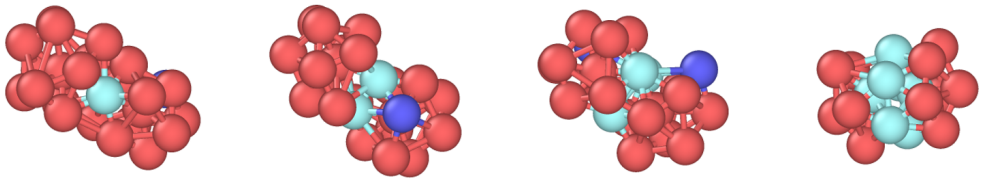
Voronoi tessellation is a method that decomposes spatial volume to each atom center and is commonly used to analyze the structure of liquids. The points closest to the center are part of the VP.[99] The Voronoi index $\langle n_3, n_4, n_5, n_6 \rangle$ describes the VP, and each n_i is the number of faces with i edges. We used the Voronoi tessellation calculation from `Ovito`.

3.2.9 Clusters

Icosahedron VP were found to be important in liquids,[71,77,78,87,88] therefore we especially consider icosahedron VPs and their clusters. An atom with $\langle 0, 0, 12, 0 \rangle$ VP index is at the center of an icosahedron VP, which is defined as the center atom and its 12 nearest neighbors. This is the smallest icosahedron cluster with 13 atoms. If any of these 13 atoms are shared with another icosahedron cluster, they are considered one cluster, and the linkage continues transiently. Common numbers of atoms shared between icosahedron VPs are one atom (vertex-sharing), two atoms (edge-sharing), three atoms (face-sharing), and seven atoms (intercross-sharing),[100,101] and example clusters are shown in Fig. 3.4. To identify and assign atoms to these clusters, we first calculate the VP index of each atom, then select the $\langle 0, 0, 12, 0 \rangle$ atoms and their nearest neighbors. For n selected atoms, we create an $n \times n$ distance matrix with zeros on i, j entries if atoms i, j are nearest neighbors and ones everywhere else. We then use agglomerative clustering from `scikit-learn` with precomputed affinity, single linkage, and 0.1 distance threshold to fit our distance matrix.[102] In our case, the distance threshold could be any number strictly greater than zero and less than one, so 0.1 was an arbitrary choice. The agglomerative clustering with these parameters links clusters transiently if they have shared atoms. We used this method because the clustering method in `Ovito` links atoms into clusters if they have shared bonds; Fig. 3.5 shows an example of the distinction. The agglomerative clustering assigns atoms to cluster-id at each trajectory timestep.

We also found the clusters which lasted along consecutive timesteps in the simulation. For the clusters at each timestep, we found the atom-ids of cluster atoms at the current and previous timestep. If at least 13 atoms in a cluster

were the same across the two timesteps, we updated the current cluster-id to match the corresponding cluster-id from the previous timestep. This algorithm was repeated for all clusters and timesteps in the simulation. We chose a minimum of 13 atoms because 13 is the smallest icosahedron cluster.



(a) (b) (c) (d)

Figure 3.4: Common icosahedral clusters that contain shared atoms (light cyan). a): One atom shared (vertex-sharing), 25 total atoms. b): Two atoms shared (edge-sharing), 24 total atoms. c): Three atoms shared (face-sharing), 23 total atoms. d): Seven atoms shared (intercross-sharing), 19 total atoms.

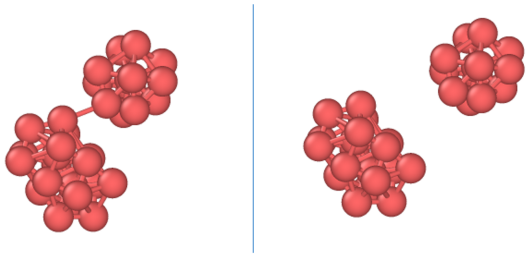


Figure 3.5: Left: Existing clustering method links by nearest neighbor bonds, resulting in one cluster. Right: Agglomerative clustering of this work clusters by shared atoms, two separate clusters.

3.2.10 Per-atom viscosity and diffusion

We developed per-atom calculation methods to approximate the clusters' effect on viscosity and diffusion. The per-atom methods decompose viscosity and diffusion contributions from each atom. Per-atom diffusion is a natural concept because diffusion coefficient is an average over atoms. We frequently calculate diffusion coefficients of different chemical elements, and researchers have taken an interest to individual particle diffusions in relation to the SER, both experimentally[103] and in simulation.[104] The per-atom viscosity here is a

practical method to approximate the effect of clusters on viscosity. We show our formulation in the remaining part of this section. Our results (Section 3.3.9) show that viscosity from normal Green-Kubo or Einstein method is the same as viscosity from the per-atom method when all atoms are included, and nearly the same when some atoms are excluded in the per-atom calculation. Therefore, the per-atom calculation is a reasonable approximation for the effect of clusters on viscosity. The per-atom viscosity calculation could also be used to determine viscosity of a liquid in contact with a solid during a simulation.

To calculate per-atom diffusion, we use `displace/atom` and `com` (center of mass) commands in `LAMMPS`. `Displace/atom` gives the x, y, and z displacements per atom from the initial configuration. We subtract the center of mass drift from the displacement at each timestep (using `pandas` in `python`). We then use the desired subset of displacements for calculating mean squared displacement and diffusion as described in Section 3.2.3.

For per-atom viscosity, we use the Einstein method as shown in Eq. 3.9.[93] Here $\tau_{\alpha\beta}$ are the off diagonal elements of the stress tensor, or $\frac{1}{V}\sum_{i=1}^{N}(mv_{\alpha,i}(t)v_{\beta,i}(t) + r_{\alpha,i}(t)f_{\beta,i}(t)), \alpha \neq \beta$ with $f_{\beta,i}$ being the force acting on particle i in direction β. Note it is possible to decompose the pressure tensor elements by atom because it is a sum over all atoms, and we use `stress/atom` compute in `LAMMPS`.[105] The per-atom stress gives the negative of the per-atom pressure tensor. For the volume, we approximate the volume per atom using the Voronoi volume and sum the per-atom volumes of the desired subset of atoms. We ran NVT simulations to collect per-atom data for diffusion and viscosity, with seven independent trials at each temperature tested. We saved snapshots of displacements and stresses per atom every $t_s \in [0.04, 0.1]$ ps apart, depending on the temperature with shorter t_s for higher temperatures. Each simulation was run for 350-1000 ps, depending on

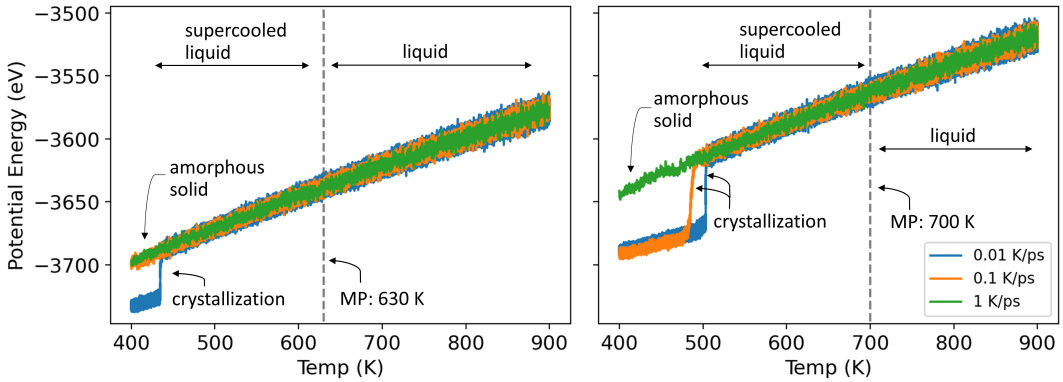
temperature with shorter simulations for higher temperatures. Following Eq. 3.9, there were 175 or 200 ensemble averages of the derivative, and the upper limit t of the integral was 2-5 ps depending on the temperature, with smaller t for higher temperatures. This is to ensure ensemble averages are statistically independent, and stress autocorrelation decays to zero within 2-5 ps, with faster decay for higher temperatures. In addition, the Einstein integral becomes nearly linear after a few picoseconds.[93] The exact value of t requires tuning and is more easily found by the Green-Kubo method for viscosity.

$$\eta = \frac{V}{2k_B T} \lim_{t\to\infty} \frac{d}{dt} \left\langle \left(\int_0^t dt' \tau_{\alpha\beta}(t') \right)^2 \right\rangle \tag{3.9}$$

3.3 Results

3.3.1 Cooling simulation

We conduct cooling simulations to observe occurrence of distinct phases at different temperatures. In direct cooling simulations, the liquid becomes supercooled liquid below the melting temperature. It is unlikely that the liquid spontaneously crystallizes at the melting point, but it may crystallize at a much lower temperature if the cooling rate is slow enough. Fig. 3.6 shows the potential energy changes with temperature for three different cooling rates.



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.6: Potential energy vs. Temperature shows presence of different phases.

At 900 K and above, the potential energies are overlapping for the different cool rates, so we did not include those temperatures in Fig. 3.6. The sudden drop in potential energy indicates the crystallization phase change. For the slowest cool rate, the crystallization occurred at 435 K for $Al_{90}Si_{10}$ and at 504 K for $Al_{95}Si_5$. For faster cool rates, the crystallization occurred at even lower temperatures or not at all. At temperatures lower than around 430 K, the atoms' mean squared displacement became near zero. If the system did not crystallize, it became an amorphous solid.

3.3.2 Melting point

We performed solid-liquid interface melting point simulations and observed the volume vs. temperature to determine approximate melting point. Figs. 3.7 and 3.8 show the volume vs. temperature for the melting simulations. Each point represents a simulation with independent initial configuration and atom velocities. In Fig. 3.7, the $Al_{90}Si_{10}$ has simulations between 612 K and 645 K with solid or liquid phases depending on initialization. Therefore we take the melting point to be at the center or 630 K. Similarly, we observe 685 K to 714 K temperatures partially melting for $Al_{95}Si_5$, and we take its melting

point to be 700 K. The melting points found by Starikov et al.,[14] using the same potential, were 740 K and 700 K for $Al_{95}Si_5$ and $Al_{90}Si_{10}$, which are close but not the same as melting points found in this work. Starikov et al. did not provide uncertainty estimates for their melting points, nor state the exact compositions used to generate the phase diagram; these factors could explain the differences in melting points. To decrease uncertainty on our calculated melting points, larger simulations with longer time and several repeated runs are required. These melting points indicate that configurations for $Al_{90}Si_{10}$ between 435 K and 630 K, and for $Al_{95}Si_5$ between 504 K and 700 K are supercooled liquids.

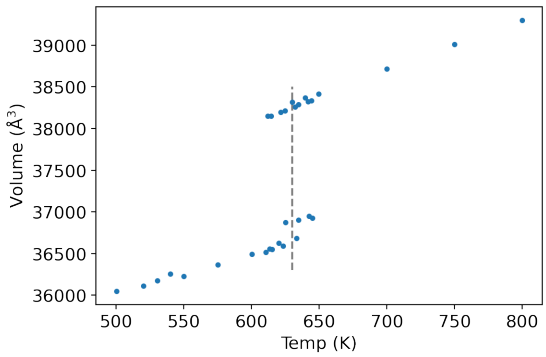


Figure 3.7: Volume vs. Temperature in NPT solid-liquid interface simulations for $Al_{90}Si_{10}$.

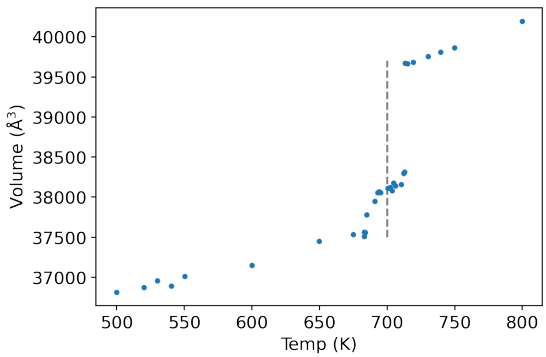


Figure 3.8: Volume vs. Temperature in NPT solid-liquid interface simulations for $Al_{95}Si_5$.

3.3.3 Diffusion

We calculated self-diffusion coefficients for the Al-Si alloy in liquid and supercooled liquid regions. Fig. 3.9 shows the diffusion coefficients. For solid and amorphous phases, the diffusion was negligible, and these lower temperatures were excluded from the plots. For both compositions, the Si diffusion was higher than Al. Eq. 3.10 shows the Arrhenius equation for diffusion, where D_0 is the diffusion coefficient at infinite temperature (pre-exponential factor), E_D is the activation energy, and T is the temperature in K.

$$D(T) = D_0 \exp\left(\frac{-E_D}{k_B T}\right) \tag{3.10}$$

The diffusion coefficients follow the Arrhenius equation for the temperature range, as shown in Fig. 3.10 along with the energy activations. In Table 3.1, we compare the diffusions of this work with an *ab initio* study.[82] The comparable diffusions are quite close except D_{Si} for $Al_{90}Si_{10}$, which may be caused by be the difference between the physical potential and *ab initio* calculation. Fig. 3.11 shows the ratio of Al to Si diffusion coefficients. The difference in diffusion between Al and Si, as measured by this ratio, increases as temperature decreases. This difference is likely caused by chemical interactions.[82] In our case, Si-Si repulsion becomes stronger as temperature decreases (Section 3.3.7), therefore Si-Si repulsion could be why $D_{\mathrm{Al}}/D_{\mathrm{Si}}$ decreases further from one as temperature decreases.

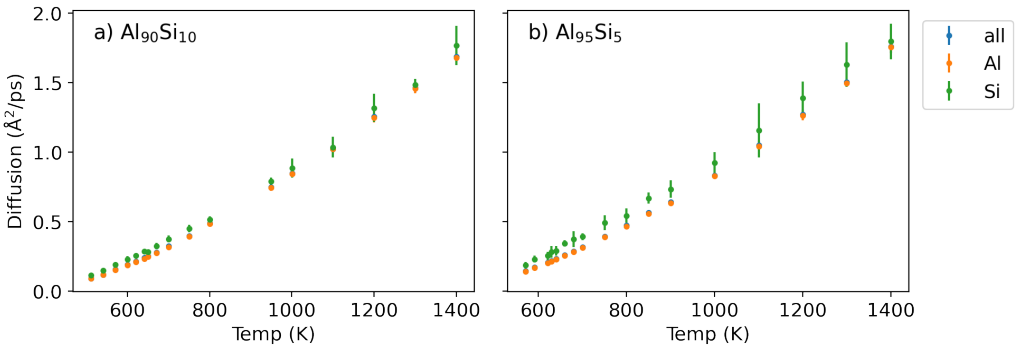
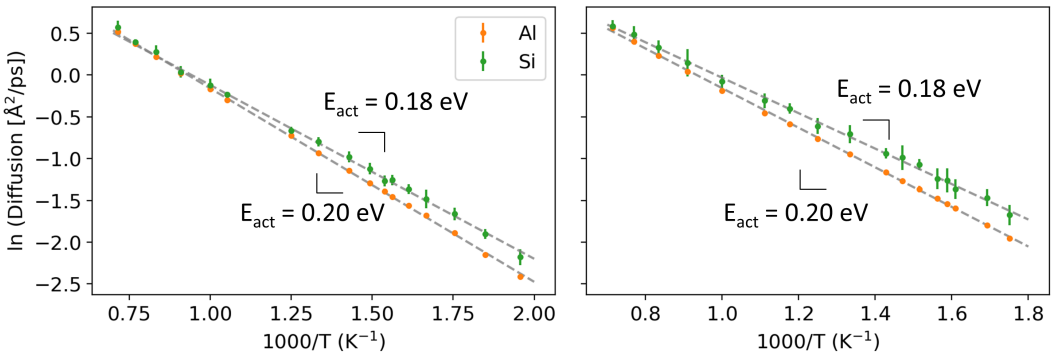


Figure 3.9: Diffusion vs. Temperature for a): $Al_{90}Si_{10}$ and b): $Al_{95}Si_5$.



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.10: Diffusion follows Arrhenius equation.

Table 3.1: Diffusion coefficients comparison with literature values[82]

	T (K)	D_{Al} (Å^2/ps)	D_{Si} (Å^2/ps)	D_{Al}/D_{Si}
$Al_{95}Si_5$	1000	0.83	0.92	0.9
$Al_{95}Si_5$[82]	1003	0.72	0.58	1.2
$Al_{90}Si_{10}$	950	0.74	0.79	0.94
$Al_{92}Si_8$[82]	960	0.65	1.0	0.65
$Al_{88}Si_{12}$[82]	960	0.77	0.87	0.89

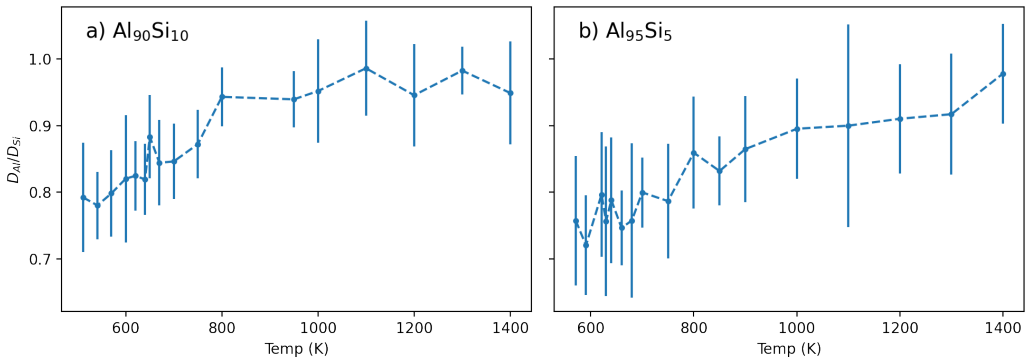


Figure 3.11: $D_{\mathrm{Al}}\,/D_{\mathrm{Si}}$ vs. Temperature for a): $Al_{90}Si_{10}$ and b): $Al_{95}Si_5$.

3.3.4 Viscosity

Viscosity is an important property that characterizes liquids. Fig. 3.12 shows the viscosities at various temperatures for $Al_{90}Si_{10}$, $Al_{95}Si_5$, and experimental viscosity for $Al_{88}Si_{12}$.[106] Viscosity increases as temperature decreases, which is expected. The MD viscosities are close to the experimental viscosities, and part of the difference is because they are different compositions. Additionally, the trend in viscosity with temperature matches well with experiment. Eq. 3.11 shows the Arrhenius equation for viscosity, where η_0 is the viscosity at infinite temperature, and E_η is the activation energy.

$$\eta(T) = \eta_0 \exp\left(\frac{E_\eta}{k_B T}\right) \tag{3.11}$$

We show the Arrhenius fit of viscosities in Fig. 3.13. For both compositions, there are two Arrhenius regions. The break between two regions occurs at 645 K for $Al_{90}Si_{10}$ and 700 K for $Al_{95}Si_5$, which are near their respective melting points, 630 K and 700 K. The two Arrhenius regions indicate that viscosity increases at a faster rate when temperature decreases.

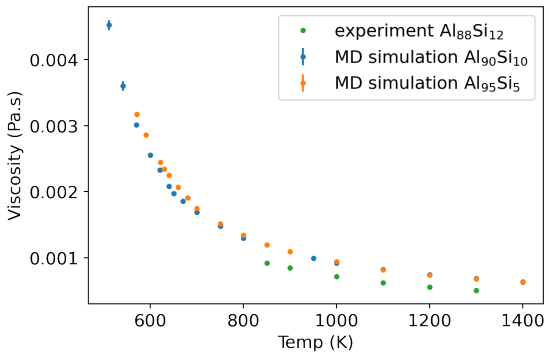


Figure 3.12: Viscosity vs. Temperature for $Al_{90}Si_{10}$ and $Al_{95}Si_5$ compared with experimental values.[106]

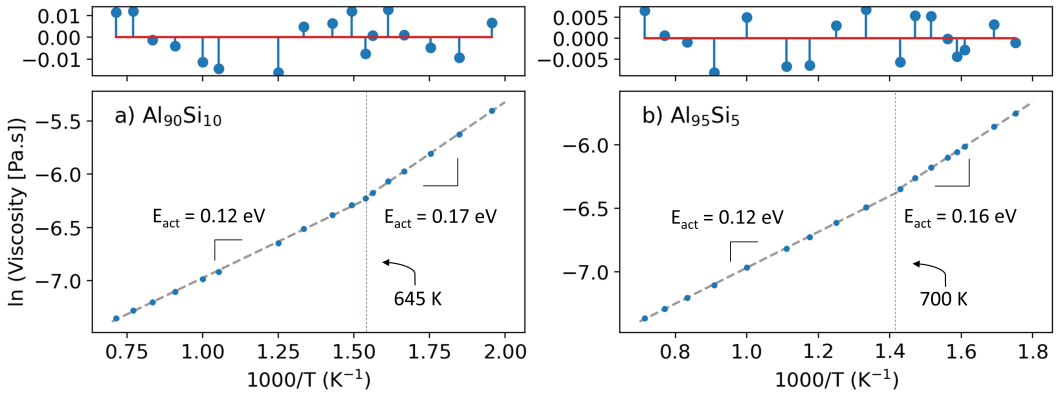
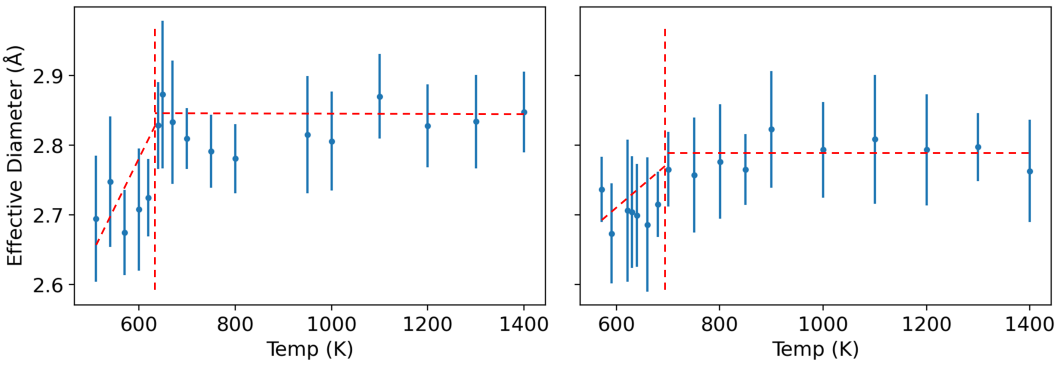


Figure 3.13: Two Arrhenius regions for Viscosity for a): $Al_{90}Si_{10}$ and b): $Al_{95}Si_5$. Top row plots are residuals of ln viscosity minus Arrhenius fit.

3.3.5 Stokes-Einstein relation

We calculate the effective diameter of the SER using viscosity and diffusion to investigate validity of the SER for the system. Fig. 3.14 shows the effective diameters for $Al_{90}Si_{10}$ and $Al_{95}Si_5$. In Fig. 3.14a, the effective diameter appears constant above 630 K, and decreases with temperature below 630 K; therefore the SER breaks down at temperatures below 630 K. In Fig. 3.14b, there are two regions with different effective diameters above and below 690 K. The results show that the SER breaks down for these Al-Si compositions near their melting points. Mathematically, the viscosity

increases too fast for SER to hold at the low temperature. In an alternative view, if diffusion was slower at low temperatures, then the SER would hold. Table 3.2 shows the pre-exponential factors and activation energies for diffusion and viscosity, and the Arrhenius fits were shown in Figs. 3.10 and 3.13. The activation energies for viscosity and diffusion (E_η and E_D) are different, η_0 appears to be dependent on temperature, while D_0 is independent of temperature in the temperature range. This indicates that $\frac{T}{\eta_0 \exp((E_\eta - E_D)/k_B T)}$ must be constant for the SER to hold. Alternatively, $\frac{\exp((E_D - E_\eta)/k_B T)}{\eta_0} \propto \frac{1}{T}$ for SER to hold. This is true in our case for $Al_{90}Si_{10}$ from 630-1400 K and $Al_{95}Si_5$ from 690-1400 K, and the respective values of $E_D - E_\eta$ are 0.079 and 0.081 eV. Fitting the FSE, the κ values are 0.97 $\pm$ 0.01 and 0.97 $\pm$ 0.01 for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. This indicates an SER deviation occurred because $\kappa \neq 1$.



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.14: Deviations in Stokes-Einstein relation for a): at 630 K and b): at 690 K.

Table 3.2: Arrhenius fit parameters for diffusion and viscosity

	T(K) range	D_0 (Å^2/ps)	E_D (eV)	η_0 (mPa.s)	E_η (eV)
$Al_{90}Si_{10}$	500 - 1430	8.5	0.197		
	500 - 645			0.099	0.168
	645 - 1430			0.24	0.118
$Al_{95}Si_5$	550 - 1430	9.0	0.203		
	550 - 700			0.13	0.158
	700 - 1430			0.23	0.122

3.3.6 Radial distribution functions

RDFs are used to examine the structure of liquids. Figs. 3.15 and 3.16 show the total and partial RDFs for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. The total RDFs at 700 and 1000 K have characteristic shape for a liquid, which verifies validity of our potential and simulations. As temperature decreases the total RDF begins to have sharper peaks, which is also expected. There is also a slight double bump in the second peak of some RDFs at the low temperature, which is sometimes observed for low temperature liquids. Origin of splitting of the second peak is an active area of research and some hypotheses are intensified icosahedral order, local translational symmetry,[107] and connections of polyhedrons of atoms.[108,109] The double bump does indicate the supercooled liquid structure begins to more closely resemble an amorphous structure as temperature decreases.[109] We also observe that first neighbor Si-Si atoms are further apart as temperature decreases, and further apart than Al-Si or Al-Al pairs.

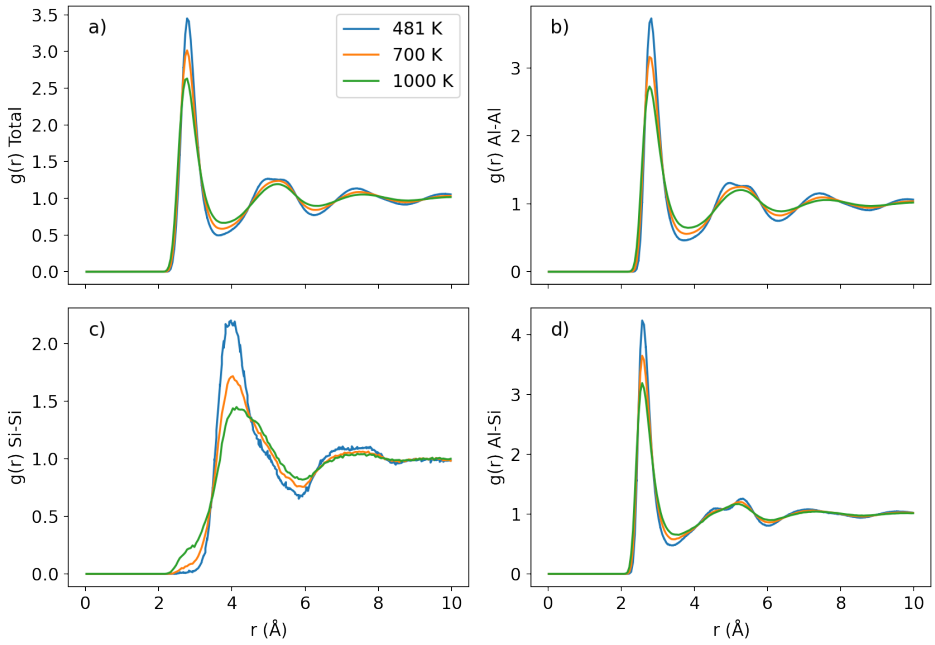


Figure 3.15: RDFs at different temperatures for $Al_{90}Si_{10}$. a): Total. b): Al-Al. c): Si-Si. d): Al-Si.

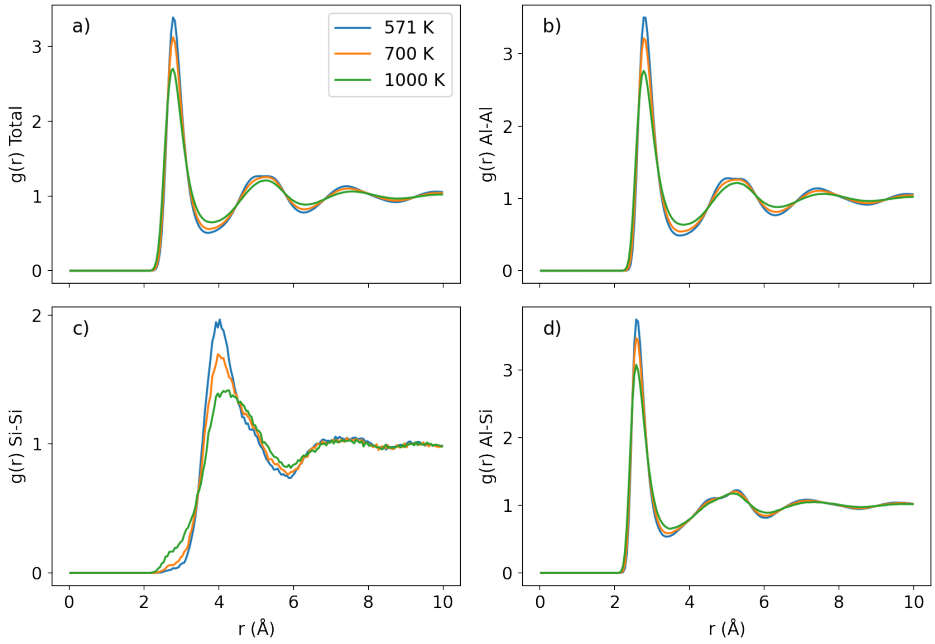


Figure 3.16: RDFs at different temperatures for $Al_{95}Si_{5}$. a): Total. b): Al-Al. c): Si-Si. d): Al-Si.

3.3.7 Coordination number and chemical short-range order

Coordination number is the number of nearest neighbors for a center atom. We examine coordination numbers and partial coordination numbers to characterize the liquid and possible chemical short-range order (CSRO). Figs. 3.17 and 3.18 show the coordination numbers vs. temperature for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. Al-Si coordination number indicates Al as the center atom. The coordination numbers for $Al_{90}Si_{10}$ were close to coordination numbers found for $Al_{88}Si_{12}$ in an *ab initio* study by Wang et al.[79] They found the Al-Si coordination number around 3.0 for the same temperature ranges, which is higher than our Al-Si coordination number around 1.0. This difference is likely due to the physical potential used vs. their *ab initio* simulation. For total, Al-Al, and Si-Al coordination numbers, the coordination numbers do not show a clear trend as the temperature changes. The Si-Si coordination number decreases as temperature decreases for both compositions, which is a change in CSRO.

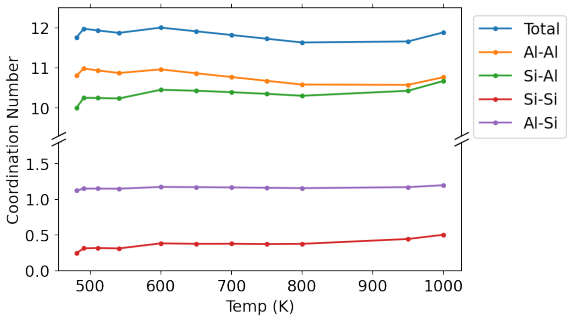


Figure 3.17: Total and partial Coordination Numbers for $Al_{90}Si_{10}$.

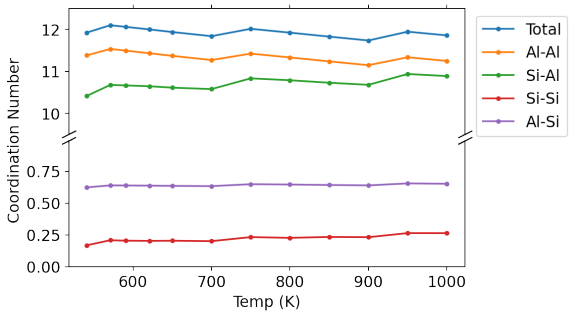


Figure 3.18: Total and partial Coordination Numbers for $Al_{95}Si_5$.

The Warren-Cowley parameter measures CSRO, and is defined in Eq. 3.12.[110,111]

$$\alpha_{ij}^{l} = 1 - \frac{N_{ij}}{c_j N_i} \tag{3.12}$$

where i is center atom species, j is surrounding atom species, l is shell, N_{ij} is i-j coordination number, c_j is concentration of j, and N_i is total coordination number around i center atoms. The value $\frac{N_{ij}}{N_i}$ is the conditional probability of j surrounding atom given i center atom, $P(j \text{ neighbor atom} | i = \text{center atom})$. If the chemical species were distributed randomly, this probability would equal the concentration of j atoms, c_j. Therefore Warren-Cowley $\alpha = 0$ represents random distribution, $\alpha < 0$ represents attractive interaction, and $\alpha > 0$ represents repulsive interaction of j surrounding i atoms. Tables 3.3 and 3.4 show the first-shell Warren-Cowley parameters at different temperatures for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. The Al-Al, Al-Si, and Si-Al parameters are near zero, meaning they are mostly random. The Si-Si parameters are largely positive, indicating that Si-Si has repulsive interaction in these compositions. In addition, the Si-Si becomes more repulsive as temperature decreases, since the magnitude of Warren-Cowley parameter increases as temperature decreases.

Table 3.3: Warren-Cowley CSRO parameter α_{ij}^1 for $Al_{90}Si_{10}$

Temp (K)	Al-Al	Si-Si	Al-Si	Si-Al
481	-0.001	0.762	0.067	-0.086
600	-0.003	0.651	0.042	-0.073
800	-0.005	0.651	0.024	-0.073
1000	-0.007	0.554	0.008	-0.062

Table 3.4: Warren-Cowley CSRO parameter α_{ij}^1 for $Al_{95}Si_5$

Temp (K)	Al-Al	Si-Si	Al-Si	Si-Al
571	-0.004	0.662	0.070	-0.040
700	-0.003	0.671	0.058	-0.040
800	-0.003	0.637	0.045	-0.038
1000	-0.002	0.582	0.030	-0.035

3.3.8 Voronoi polyhedrons

We obtained the Voronoi indices of each atom in the frames of the simulations. The most common indices are shown in Figs. 3.19 and 3.20 for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. The perfect icosahedron is $\langle 0, 0, 12, 0\rangle$, distorted icosahedrons are $\langle 0, 1, 10, 2\rangle$ and $\langle 0, 2, 8, 2\rangle$.[112] Our data also had $\langle 0, 1, 10, 3\rangle$ and $\langle 0, 2, 8, 4\rangle$ indices which appear to be distorted icosahedrons. Distorted FCC polyhedrons are $\langle 0, 3, 6, 4\rangle$, $\langle 0, 3, 6, 5\rangle$, $\langle 0, 4, 4, 6\rangle$,[112] and our data also had $\langle 0, 4, 6, 3\rangle$ and $\langle 0, 4, 6, 4\rangle$ indices which are similar to distorted FCC polyhedrons. In Figs. 3.19 and 3.20, the Al atoms for all indices except $\langle 0, 4, 6, 3\rangle$ and $\langle 0, 4, 6, 4\rangle$ increased as temperature decreased. The change in Si fraction is not as dramatic, but there is a slight increase in fraction of Si atoms with $\langle 0, 2, 8, 2\rangle$ and $\langle 0, 3, 6, 4\rangle$ indices as temperature decreases. A larger fraction of Al than Si atoms have $\langle 0, 1, 10, 2\rangle$, $\langle 0, 2, 8, 4\rangle$, $\langle 0, 3, 6, 4\rangle$, $\langle 0, 3, 6, 5\rangle$, and $\langle 0, 4, 4, 6\rangle$ indices.

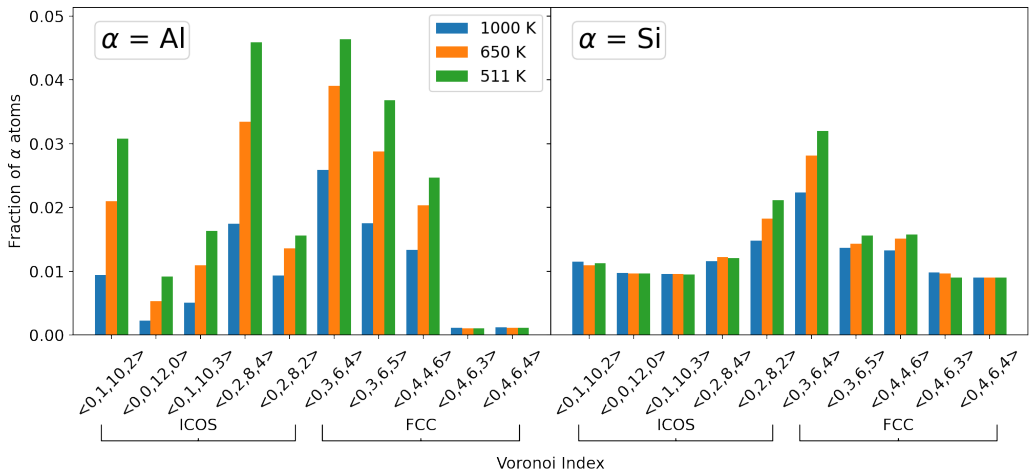


Figure 3.19: Common Icosahedron (ICOS) and FCC-like Voronoi polyhedrons for $Al_{90}Si_{10}$.

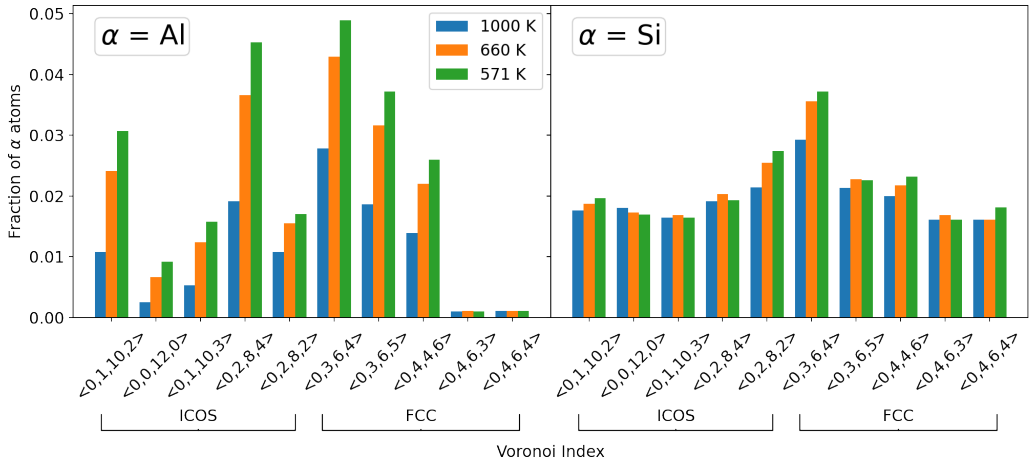
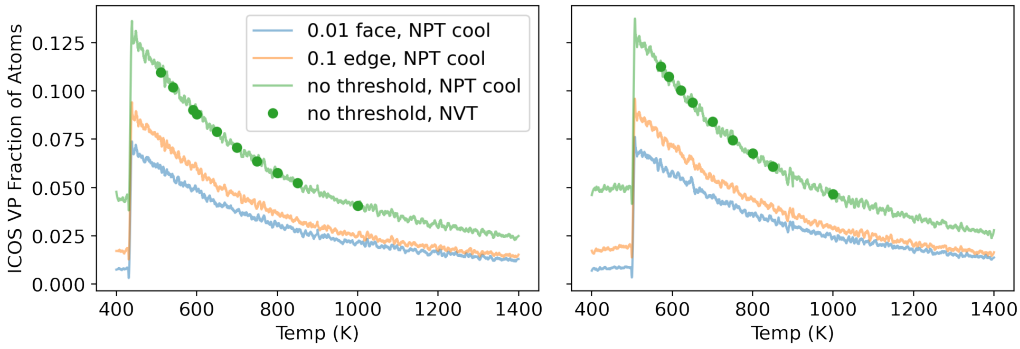


Figure 3.20: Common ICOS and FCC-like Voronoi polyhedrons for $Al_{95}Si_5$.

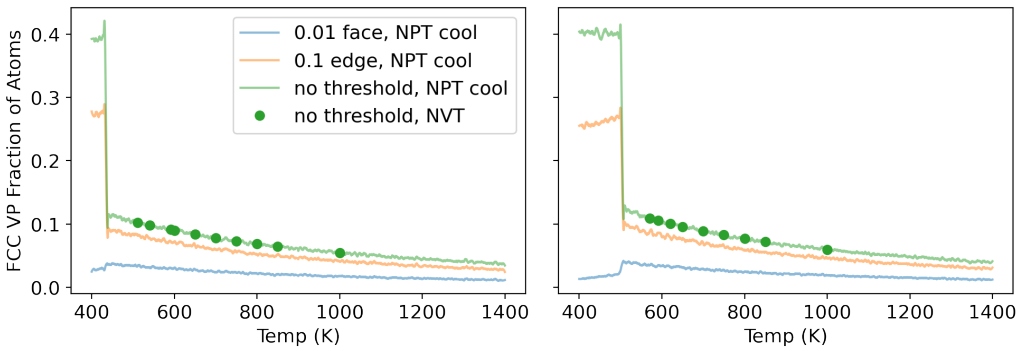
We observe how the fraction of icosahedron (ICOS) and FCC-like VP change as temperature decrease. In Fig. 3.21, we show the sum of all ICOS indices from Fig. 3.19 vs. temperature. Fig. 3.21 includes data from the NPT cooling simulation with 0.01 K/ps cooling from 1400 to 400 K. This data was smoothed using Savitzky-Golay filter.[113] The three lines indicate different sets of hyperparameter settings for Voronoi tessellation: minimum

edge threshold 0.1 Å, minimum face threshold 1%, and no thresholds. There are differences between Voronoi index classification depending on the hyperparameter settings, but the trends are the same in Fig. 3.21. In all other results, "no thresholds" were used unless specified. Fig. 3.21 also includes the data from the NVT simulations which match the smoothed NPT cooling simulation data. Fig. 3.21 shows a smooth increase fraction of ICOS VPs as temperature decreases, with a sudden decrease at the crystallization phase change. Fig. 3.22 shows the sum of all FCC indices from Fig. 3.19 vs. temperature. The NPT signal-filtered data matches with the NVT data. The three hyperparameter settings show the same trend in the liquid and supercooled liquid regions, but differ at the crystallization. This indicates Voronoi tessellation is sensitive to FCC characterization if the FCC lattice is not perfect. Fig. 3.22 shows the fraction of FCC VPs increases as temperature decreases in the liquid and supercooled liquid region.



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.21: Fraction of ICOS Voronoi polyhedron atoms vs. Temperature.



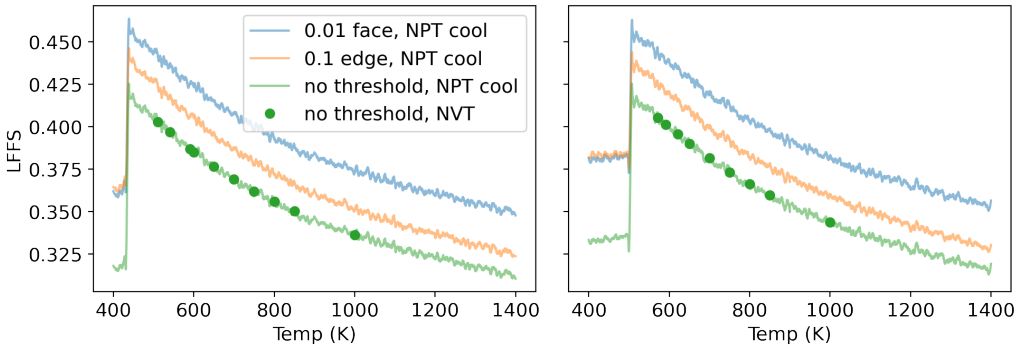
(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.22: Fraction of FCC Voronoi polyhedron atoms vs. Temperature.

Five-fold symmetry in liquids has been found to be important for dynamic slowdown.[78] We calculate the average local five-fold symmetry (LFFS), shown in Eq. 3.13, where n_i are number of i faced edges of the Voronoi polyhedron.[71]

$$d_5 = n_5 / \sum_i n_i \tag{3.13}$$

Fig. 3.23 shows the LFFS change with temperature. The same trend occurs for the three hyperparameter settings of Voronoi tessellation. The LFFS increases as temperature decreases until a steep drop at the crystallization. However the LFFS increases steadily without a sudden acceleration at any temperature in Al-Si, while a sudden acceleration was observed in Al-Fe liquid alloys.[71] This indicates that LFFS is not a universal indicator of Stokes-Einstein deviation.



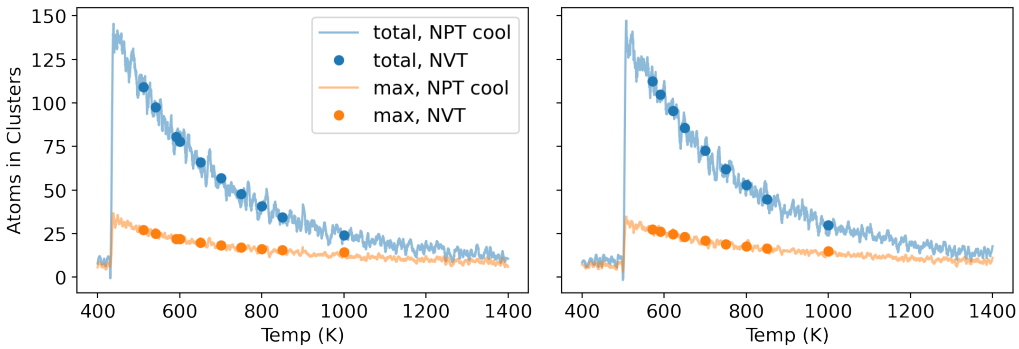
(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.23: Local five-fold symmetry vs. Temperature.

We assigned atoms to clusters surrounding center atoms with the $\langle 0, 0, 12, 0 \rangle$ icosahedron index, as described in Section 3.2.9. Fig. 3.24 shows number of atoms belonging to clusters and number of atoms in the largest cluster vs. temperature. The lines are signal-filtered data from NPT cooling simulation, and points are data from NVT simulation. The results from the two types of simulations are consistent. Fig. 3.24 shows an increase in the atoms in clusters and the size of the largest cluster as temperature decreases until the crystallization phase change. The expectation of the size of the largest cluster is around 40 atoms, just before the crystallization. Also the maximum expectation of total atoms in clusters occurs just before crystallization and is 140 atoms, which is 12.7% of atoms.

We also analyze the life of the clusters, or time duration that a cluster lasts in the simulation. Cluster assignment over time is described in Section 3.2.9. Fig. 3.25 shows the percent of clusters lasting longer than 1 ps vs. temperature. At 800 K, almost no clusters last longer than 1 ps. As temperature decreases, more clusters form, and more of the clusters last longer than 1 ps. One picosecond is significant because it is 1,000 simulation steps and shows the clusters are real. At the same temperature, $Al_{95}Si_5$ has a slightly larger fraction of long lasting clusters than $Al_{90}Si_{10}$. For $Al_{90}Si_{10}$,

7.5% of clusters at 510 K last at least 1 ps, and for $Al_{95}Si_5$, 4.3% of the clusters at 570 K last at least 1 ps. At these supercooled temperatures before crystallization, the vast majority of clusters still last shorter than 1 ps. (The cutoff time was 1 ps because the analysis is memory intensive.)



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.24: Total atoms in clusters and largest clusters vs. Temperature.

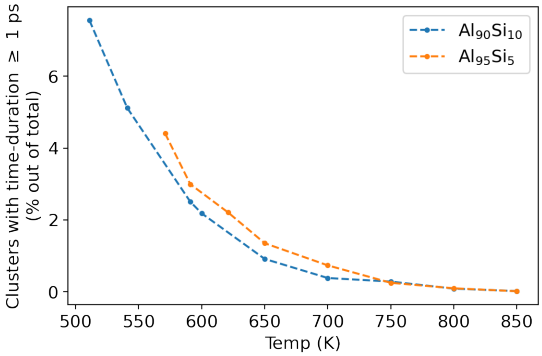
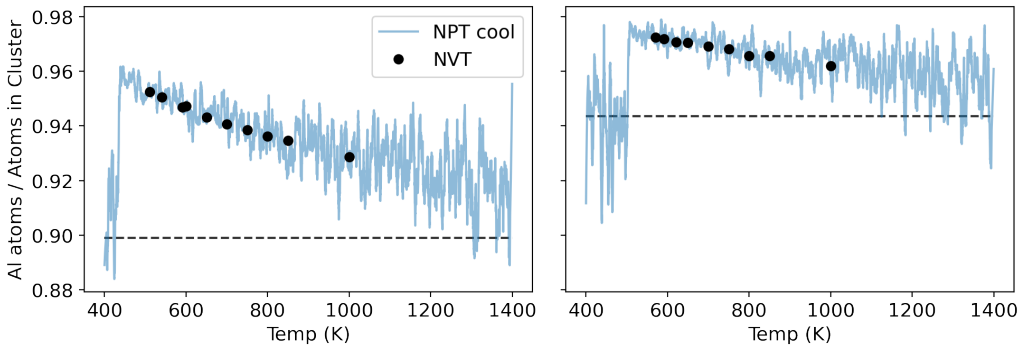


Figure 3.25: Percent of clusters that last longer than 1 ps vs. Temperature.

We examine the chemical composition of the clusters. Fig. 3.26 shows the fraction of Al atoms in clusters vs. temperature. The NPT cooling simulation data is signal-filtered but still quite noisy, and NVT data matches the NPT data. The black dashed lines are the overall concentration of Al. In Fig. 3.26, the fraction of Al in clusters is higher than overall Al concentration, which indicates Al atoms are more likely to be in a cluster than Si atoms. The fraction of Al in clusters increases as temperature decreases, showing segregation of Al

into clusters becomes stronger as temperature decreases. The weak repulsion of Si-Si could explain why Si atoms are less likely to be in the icosahedral clusters.



(a) $Al_{90}Si_{10}$ (b) $Al_{95}Si_5$

Figure 3.26: Fraction of Al atoms in clusters is higher than Al concentration.

3.3.9 Cluster effects on viscosity and diffusion

We use the per-atom methods described in Section 3.2.10 to isolate the effect of clusters on diffusion and viscosity. If an atom was in an icosahedral cluster in two adjacent timesteps, we separate its contribution from the remaining atoms' data at that timestep. Fig. 3.27 shows the per-atom diffusions, and they are the same as diffusions in Fig. 3.9, which were calculated from `LAMMPS msd` command. The per-atom diffusions with and without atoms in clusters are the same. This indicates that icosahedral clusters do not significantly alter the diffusivity. It is possible that a cluster diffuses as one entity, or each atom goes in and out of clusters over time, therefore the long-time diffusivity is the same for atoms within and outside of clusters.

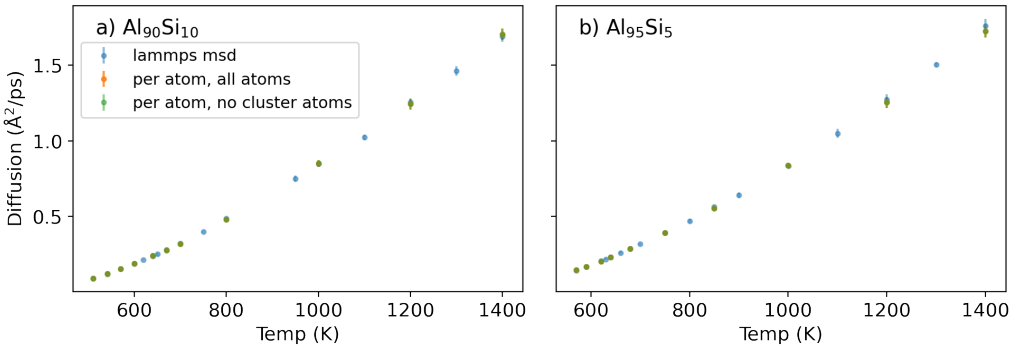


Figure 3.27: Per-atom diffusion vs. Temperature for a): $Al_{90}Si_{10}$ and b): $Al_{95}Si_5$. Diffusions for atoms inside and outside clusters are the same.

We examine how clusters affect the viscosity. Fig. 3.28 shows the per-atom and Green-Kubo viscosities. The per-atom viscosities including all atoms align with the Green-Kubo viscosities from Fig. 3.12. The uncertainty using Einstein method is higher than Green-Kubo for viscosity, which is an expected difference between the two methods.[93] The viscosities excluding cluster atoms are always lower than the viscosities with all atoms. This indicates that clusters increase the viscosity, which is reasonable because we expect clusters to increase a fluid's autocorrelation, and SACF decay time. Intuitively, clusters should thicken a fluid. The differences in all-atom and no-cluster viscosities increase as temperature decreases, because more atoms are in clusters as temperature decreases. The per-atom method allows us to isolate the effect of individual atoms and subgroups of atoms in the liquid and shows the intuitive result that clusters increase viscosity in the liquid alloy.

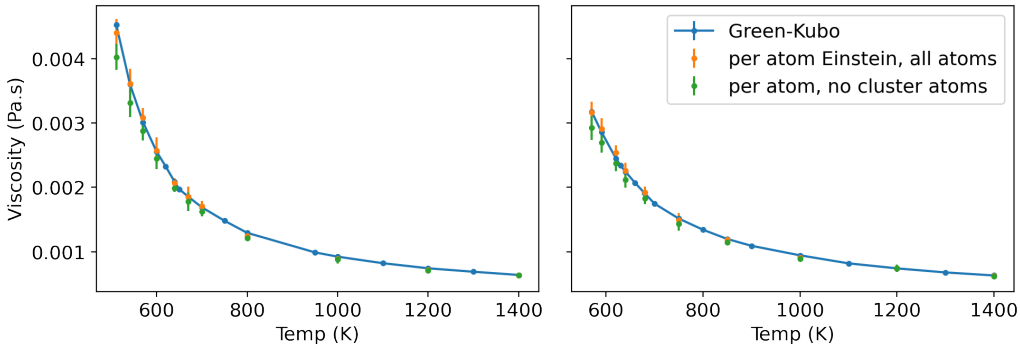


Figure 3.28: Per-atom viscosity vs. Temperature. Per-atom Einstein viscosities match with Green-Kubo viscosities. Per-atom viscosity decreases when atoms in clusters are excluded, indicating that clusters increase viscosity.

We want to determine if the difference in viscosities with and without clusters can explain the SER deviation. Figs. 3.29 and 3.30 show the effective diameter vs. temperature for $Al_{90}Si_{10}$ and $Al_{95}Si_5$, respectively. Figs. 3.29a and 3.30a show the effective diameter calculation using per-atom viscosities and diffusions with cluster atoms removed, while Figs. 3.29b and 3.30b show the per-atom viscosities and diffusions with all atoms. In Figs. 3.29b and 3.30b, the SER deviations are expected for the entire system, and the same deviations are observed in Fig. 3.14. However, in Figs. 3.29a and 3.30a, there is no longer a clear SER deviation when the cluster atoms' contributions are removed. We also fit the FSE to the data. For data in Figs. 3.29a and 3.29b, $\kappa = 1.00 \pm 0.02$, and $\kappa = 0.96 \pm 0.02$, respectively. For Figs. 3.30a and 3.30b, $\kappa = 0.99 \pm 0.02$, and $\kappa = 0.95 \pm 0.02$, respectively. The κ values quantitatively show that SER breaks down when cluster atoms are included but holds when cluster atoms are excluded. This indicates that the increase in viscosity attributed to clusters can explain the SER deviation in liquid Al-Si.

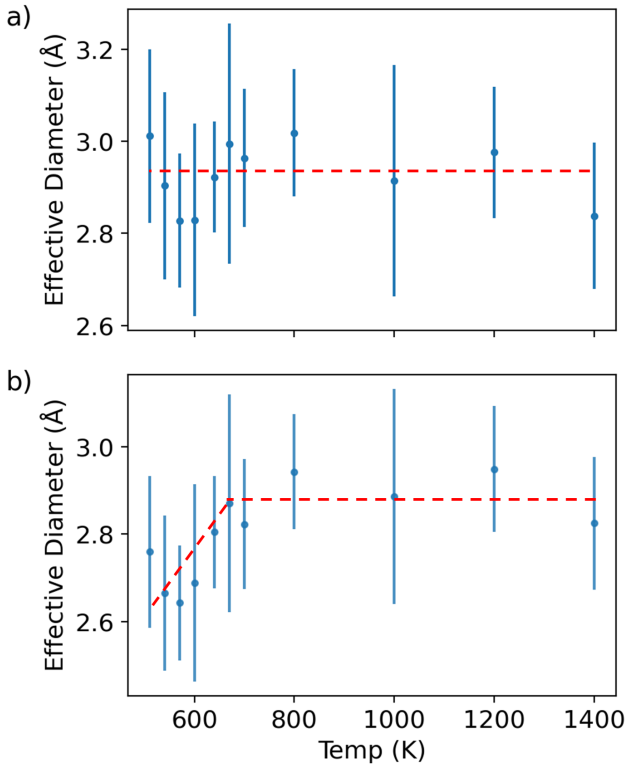


Figure 3.29: Effective diameter vs. Temperature for $Al_{90}Si_{10}$ using per-atom viscosities and diffusion coefficients. a): Atoms in clusters are removed. b): Atoms in clusters are included.

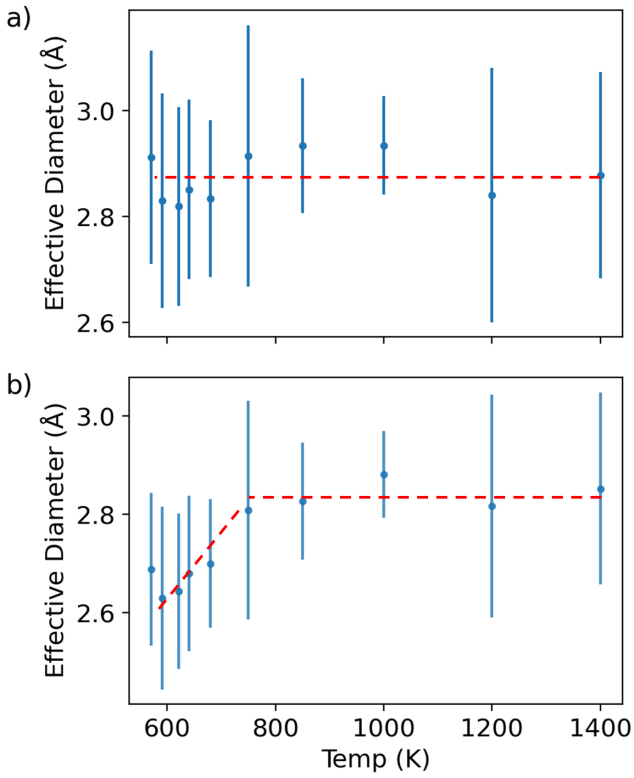


Figure 3.30: Effective diameter vs. Temperature for $Al_{95}Si_5$ using per-atom viscosities and diffusion coefficients. a): Atoms in clusters are removed. b): Atoms in clusters are included.

3.4 Conclusions

We simulated liquid $Al_{90}Si_{10}$ and $Al_{95}Si_5$ at different temperatures to investigate the Stokes-Einstein relation (SER) and local structure. Diffusion follows the Arrhenius equation for 1400 K to the supercooled temperatures just before spontaneous crystallization. Two distinct Arrhenius equations can characterize the viscosities at different temperature ranges, with the breaks near the melting points of the compositions. We found the SER holds from 1400 K and below to the melting point and deviates in the supercooled temperature region. The actual viscosity is higher than predicted by SER, or alternatively, the actual diffusion is larger than predicted by SER. In examining chemical short-range order, we found weak Si-Si repulsion at the

low Si compositions tested. The icosahedral and FCC-like Voronoi polyhedrons atoms increase as temperature decreases, with a jump change at crystallization. We assigned atoms into clusters surrounding icosahedral Voronoi atoms using agglomerative clustering. We found real and significant clusters with longest lasting clusters of 8 ps. The fraction of long lasting clusters, size of clusters, and fraction of atoms in clusters all increase as temperature decreases in the liquid and supercooled regions. We find more Al atoms in clusters and slower diffusion for Al compared with Si. These two trends become stronger and move in the same direction as temperature decreases. We developed a method to calculate viscosity and diffusion contribution per-atom, and showed that clusters increase viscosity but have a minimal impact on diffusion. Furthermore, we showed that the viscosity increase attributed to clusters leads to a SER deviation.

4 Uncertainty Quantification in Machine Learning and Nonlinear Least Squares Regression Models

4.1 Introduction

Machine learning (ML) models are used in many fields of science and engineering. They can decrease computational time, make predictions and forecasts, and improve insights of complex and high dimensional datasets. Models are more useful when they provide a prediction with its uncertainty, and in some applications it may be critical to provide a reliable uncertainty estimate.[114,115] The uncertainty estimate should also help identify input data regions that lead to extrapolation. Overall, ML models can be used more reliably when robust uncertainty quantification is available. In the simple case of low dimensional linear regression, an analytical prediction interval is available[116] and can be used to calculate the uncertainty intervals in many statistical software packages. The analytical prediction interval for linear regression requires $(\mathbf{X}^T\mathbf{X})$ to be invertible, where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the design matrix with n data points and d feature dimensions. In general for more complex (nonlinear) models and higher dimension datasets, analytical prediction intervals do not exist and alternative methods are required.

Uncertainty quantification methods for ML models are an active area of research.[117–119] Some common methods are model ensembling, training models with built in uncertainty such as Gaussian process (GP) regression, and quantile regression.[120,121] If we already have a model with trained parameters, we may want to avoid training a different model type or additional model ensemble. In these cases, this chapter presents a simple

uncertainty quantification method for parameterized models trained on minimizing the summed squared error between a model and a dataset. The method exploits automatic differentiation to calculate the Hessian of the loss function based on summed squared errors, and provides an uncertainty estimate which depends on the prediction point, model training data, and model itself. We show how the uncertainty can identify if extrapolation is occurring and aids in dataset selection for an example application of molecular simulation. In the remaining sections, we review the background on uncertainty quantification and an application of ML models for molecular simulation, describe the aforementioned delta method, and show its use in an example neural network that predicts energies from atomic structures.

4.1.1 Uncertainty quantification methods

Uncertainty quantification methods for models include bootstrap, ensembling, using model-specific uncertainty, and the delta method. Bootstrap, ensembling, and the delta method can be used for parametric models and neural networks (NNs). Bootstrap uncertainty is based on statistical theory, has some different variations, and requires training multiple models on different bootstrap samples of the data or residuals.[122–124] With the different models, the uncertainty on predictions can be estimated.

Ensemble methods require training multiple models on the entire dataset, and the different models give the uncertainty.[125–127] The uncertainty estimate quantitatively improves as the ensemble size increases, so the optimal number of ensemble models is unknown and must be user determined.[128] Some studies used bootstrap uncertainty for NN models and showed that it gave more reliable uncertainties than the delta method,[129,130] but training multiple models is computationally expensive and time consuming.

Model-specific uncertainty include Gaussian process regression,[131] dropout for NNs,[132] and Bayesian NNs.[133–135] Obtaining uncertainties in this way limits the possible mathematical forms of the model. Dropout and Bayesian NNs are also more difficult to train than standard NNs. Section 4.1.2 describes specific instances in which ensembling, bootstrap, and Gaussian process methods were used for molecular simulation. In another method, the posterior of model parameters can be approximated by the Laplace approximation, which is the second order Taylor series expansion around the optimal model parameters.[136,137] Hence the Hessian of the log likelihood contains relevant information about uncertainty.

This chapter focuses on the delta method, which also uses the Hessian of the log likelihood. The method is based on linearly approximating the model and uses an estimate of the standard error of model parameters assuming maximum likelihood estimation (MLE). Therefore, the method applies to models with parameters trained by minimizing squared error, and the model structure could be a simple linear regression to complex nonlinear regression including NNs. Different variants of the method use approximations of the Hessian, and experiments tested the delta method on NN models.[129,138–142] We further describe the method in Section 4.2 with theoretical details in Appendix D.

4.1.2 Addressing uncertainty in molecular simulation

We examine uncertainty quantification using the example application of molecular simulation, an area which has benefited from ML. Here we describe the background of ML for molecular simulation. Simulations allow researchers to obtain materials' physical properties and quickly screen materials. Molecular dynamics (MD) and Monte Carlo simulations require a

model of the potential energy surface (PES) which predicts energies and forces from atomic configurations. The options for the PES model include first principles methods such as density functional theory (DFT), physical potentials, and ML potentials. ML potentials aim to achieve the high accuracy of DFT at a significantly faster computational time. ML potentials are also more systematically improvable than physical potentials.[17] Many studies have successfully used ML potentials in simulations.[35–38]

Uncertainty quantification is useful for ML potentials. Commonly used ML potentials such as NNs will usually unreliably extrapolate on inputs much different from their training data. A likely consequence of extrapolation during a molecular simulation is wrong or unphysical results. The best ways to select enough of the relevant training space are nonobvious, since the space of atomic structures is often large, not well understood, and not possible to enumerate. Further, atomic structures are translated into fingerprints which are high dimensional and less human-interpretable than the original atomic configurations. Hence, we require a method to determine the uncertainty of a prediction from an ML potential, and the quantitative uncertainty helps us avoid extrapolation and identify sparse regions in the training dataset.

Current methods developed to address uncertainty are ensemble of potentials, on-the-fly methods, and using ML models with built-in uncertainty. Ensemble methods independently train two or more ML potentials, and check for agreement between them. In Behler's approach, they trained NNs with different architectures, and atomic structures whose predictions' differ significantly across NNs are added to the training set.[33,44,143] Peterson et al. trained an ensemble of 50 NN potentials and found that ensemble spread was a good indicator for prediction error across

the space.[144] Smith et al. also used ensemble disagreement to approximate prediction error and select a small training set.[49,145]

In MD simulations, on-the-fly methods use an ML potential augmented with quantum mechanical (QM) calculations.[146] There is a query if the ML prediction can be used. If it fails, a QM calculation is run and added to a database, and the ML model can be retrained. A simple query is if the fingerprint is out of the minimum and maximum bounds in the current database.[42] This is a minimum requirement that the ML model is not extrapolating; however, guaranteeing that the fingerprint is within bounds of training data does not guarantee a low error.[147]

Another approach is training Gaussian process regressions (GPRs)[47,146] or other ML models with built-in uncertainty estimates. Vandermause and Xie et al. used GP uncertainty to train potentials on-the-fly.[148,149] Many ML models, such as NNs, do not have theoretical guarantees for uncertainty of a prediction. Perturbation of NN weights could provide some range of uncertainty.[150] Other work used dropout in NN training as a Bayesian approximation and thereby calculating uncertainties for interatomic potentials.[151] Janet et al. used the distance in values of the last layer of NNs (or latent space) as an uncertainty measure.[152] Tran et al. compared GP, Bayesian NN, dropout NN, and ensembles of different NN structures and found more conservative uncertainties for GP and overconfident uncertainties for Bayesian NN, dropout, and NN ensemble.[153]

Musil et al. compared GP, ensembling with random subsets of the data, and bootstrap methods for obtaining uncertainties of predicting formation energies on molecular datasets.[154] They found that random sampling was easier to implement than bootstrapping, computationally faster than GP for uncertainty estimates, and matches the true error and uncertainty from GP.

Li et al. trained NN potentials with different NN structures (number of nodes), weight initialization, and learning rates, and compared the resulting prediction accuracies.[155] Their work showed a quantitative uncertainty arising from some NN hyperparameters, but it required training several NN potentials for a new system, and does not provide confidence or prediction intervals. In an alternative approach, Botu et al. fitted an empirical function to an uncertainty estimate as a function of fingerprint distance between input and reference training fingerprints.[156] Their uncertainty estimation approach requires a larger training set size.

Overall, there is not a clear consensus on the best uncertainty quantification method, and its selection usually depends on the model form used, e.g. built-in uncertainty from GPR or ensembles when using NNs. The delta method provides a simple alternative for providing quantitative uncertainty when a pretrained model exists, without the necessity of training additional models.

4.2 Methods

The delta method applies to regression problems of a model g with parameters θ. The residuals of model prediction are assumed to be Gaussian distributed and centered around zero. We assume the model parameters $\hat{\theta}$ were obtained by minimizing a function of the summed squared errors, although the method can be extended to maximize a posteriori estimation and cross-entropy loss for classification tasks.[157] We obtain an approximate standard error of a model prediction $g(\hat{\theta}, x)$ by using a Taylor series approximation and an approximate standard error of $\hat{\theta}$. Suppose that $\frac{\partial g(\hat{\theta},x)}{\partial \hat{\theta}}$

is nonzero, then the standard error of $g(\hat{\theta}, x)$ for a point x is given in Equation 4.1 using the delta method.

$$\text{se}(g(\hat{\theta}, x)) \approx \sqrt{\frac{\partial g(\hat{\theta}, x)}{\partial \hat{\theta}}^T I_n^{-1} \frac{\partial g(\hat{\theta}, x)}{\partial \hat{\theta}}} \tag{4.1}$$

where $\frac{\partial g(\hat{\theta}, x)}{\partial \hat{\theta}}$ is the gradient vector of the model with respect to parameters at the point x for which we are calculating uncertainty, and I_n is the Fisher information matrix, defined as the expectation of the Hessian of the negative log likelihood. The Fisher information is related to the Hessian of the loss, usually the sum of squared errors, by a scaling factor. Equation 4.1 shows that the model prediction of the standard error is a function of the training data, model, and point for which the uncertainty is calculated.

For small to medium models, the delta method is faster and easier to implement compared to ensembling, and the Hessian and gradients of the model are readily obtained with automatic differentiation that is included in most machine learning packages. To demonstrate the ease of use, we show a simple code example using the `autograd`[158] package in Section 4.2.2. The delta method is limited by model size since the Hessian will be $m \times m$ where m is number of parameters, and the Hessian needs to be inverted. For very large models and NNs, inverse Fisher information approximations have been proposed. Ritter et al. implemented Kronecker-factored Hessian for a network with around 2 million parameters,[136] and Nilsen et al. proposed and implemented eigenvalue spectrum Hessian approximation on networks with around 100,000 parameters.[157] For even larger networks, other uncertainty quantification methods would likely need to be used.

The uncertainties are calculated after the model has finished training, and the Fisher information inverse only needs to be calculated once per

model and training dataset. In previous tests, a model with 861 parameters and 1,900 training data points required around five minutes to calculate the inverse Fisher information with Intel Core i7-7820HQ CPU @ 2.9GHz using `autograd`. Using more modern automatic differentiation frameworks is expected to be faster. Calculating the uncertainties after obtaining the inverse Fisher information requires much less time. Theoretically, calculation of the Fisher information matrix scales quadratically with number of parameters and linearly with number of training data points.

The quality of standard errors calculated using the delta method depends on the fit of the model. We found that well fitted models have better uncertainty measures, and our assumptions required residuals to be independent, and identically distributed normal around zero. Poorly fitted models have uncertainty measures that are less quantitatively accurate.

4.2.1 Practical modifications to the inverse Fisher matrix

There are a few steps or best practices to modify how the inverse Fisher information matrix is computed.

1. We start with H, the Hessian of the sum squared errors loss function. For some models, such as NNs, the Hessians of the loss functions with respect to parameters are often nearly singular with some eigenvalues much larger than the others,[159,160] and the optimal parameters may be at a saddle point.

2. Add a small number ϵ to Hessian diagonal. Adding ϵ to the diagonals makes the matrix better conditioned for taking its inverse. ϵ should be larger in magnitude than the most negative eigenvalue. We used $\epsilon = \max(1e{-}5, 1.05 \cdot \text{abs}(\lambda_{\min}(H)))$, where $\lambda_{\min}(H)$ is the smallest eigenvalue of H. Modifying the Hessian of objective function with respect to NN

parameters has been suggested in literature and is justified because the top eigenvalues are a few orders of magnitude larger than the other eigenvalues.[136,160,161] Also note the Hessian conditioning suggests that the number of parameters of the NN is much larger than the actual degrees of freedom of the NN.

3. We take the Moore-Penrose pseudoinverse $(H + \epsilon I)^{-1}$. If the inverse exists, which is most cases following step 2, it is the same as the true inverse.

4. Multiply $(H + \epsilon I)^{-1}$ by a scaling factor α. This is done to calibrate the uncertainties to be near the residuals. We set α to be mean squared error (MSE) in most cases. To select α, we suggest trying $n^{\beta} \cdot MSE$ where n is number of training data points and β is any nonnegative number, but usually in range [0, 1]. The α should be chosen so that uncertainties are the same order of magnitude as the residual errors for the training dataset.

5. Force the scaled inverse $P := \alpha(H + \epsilon I)^{-1}$ to be positive semi-definite. For eigendecomposition $P = Q\Lambda Q^{-1}$, the closest positive semi-definite matrix in terms of Frobenius norm is $Q \max(\Lambda, 0) Q^{-1}$, where max is element-wise max.[162]

The final inverse Fisher information I_n^{-1} used in Equation 4.1 is $Q \max(\Lambda, 0) Q^{-1}$.

4.2.2 Code example

We show a simple code implementation of the delta method in Listing 1 using the `autograd` package.[158] Note that obtaining the required gradients is a single line for the Hessian (line 23) and gradients (line 28), demonstrating

the ease of automatic differentiation. Similar codes would apply in `PyTorch`[163] or other machine learning packages. In this simple example we fit a quadratic function to some slightly noisy data, and show the resulting confidence intervals on the fit. The Hessian in this case was well-conditioned, so the modifications described above were not necessary.

```
import autograd.numpy as np
from autograd import elementwise_grad, hessian
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from scipy.stats.distributions import t

x = np.array([0.1, 0.3, 0.5, 0.7, 0.9])           # {x, y} data
y = np.array([0.0, 0.1, 0.3, 0.5, 0.8])

def g(theta, x):                                   # suppose we want to fit function g
    '''function with parameters theta'''
    return theta[0] * x**2 + theta[1] * x + theta[2]

def sse(theta):
    '''Summed squared error objective function'''
    return np.sum((g(theta, x) - y)**2)

initial_guess = np.array([0.1, 0.5, 0.2])
sol = minimize(sse, initial_guess)                 # minimize sse
theta = sol.x
ypred = g(theta, x)

h = hessian(sse)(theta)                            # obtain Hessian of sse using autograd
p = sse(theta) / len(x) * np.linalg.pinv(h)        # inverse and scale Hessian

uncerts = []
for xi in x:
    gprime = elementwise_grad(g, 0)(theta, xi)     # obtain gradient using autograd
    uncerts += [np.sqrt(gprime @ p @ gprime)]      # delta method
uncerts = np.array(uncerts)

tval = t.ppf(0.975, len(x))                        # t-value

plt.plot(x, y, 'o')                                # plot the data, fit, confidence intervals
plt.plot(x, ypred)
plt.plot(x, ypred + tval * uncerts, '--r')
plt.plot(x, ypred - tval * uncerts, '--r')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['Data', 'Prediction', '95% confidence'])
plt.savefig('simple-code-ex.png')
```

Listing 1: Autograd example of the delta method.

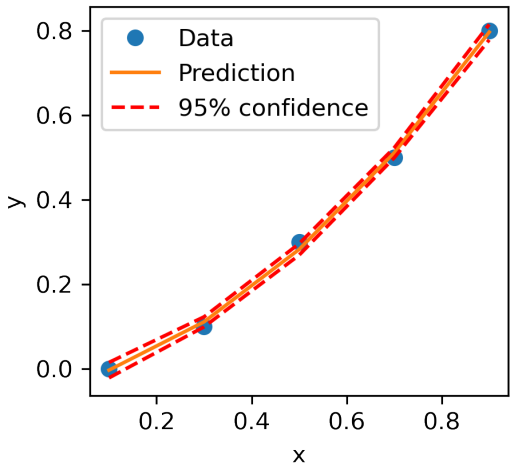


Figure 4.1: Result from Listing 1.

This simple example shows all the pieces of the delta method. There is data, and a function (line 10) with parameters that are fitted to the data. The regression here is done by optimization (line 19); this problem is linear and could be solved analytically, but we show the optimization approach for generality. We used automatic differentiation to obtain the Hessian (line 23) and gradient of the function (line 28) with respect to the parameters. The rest is conventional linear algebra.

In calculating the t-value (line 32), technically the degrees of freedom should be used instead of number of data points. However for large NNs, the effective degrees of freedom is much smaller than the number of model parameters, and determining NN degrees of freedom is an active area of research.[164–166] For simplicity, we used the number of data points to estimate the t-value throughout our results.

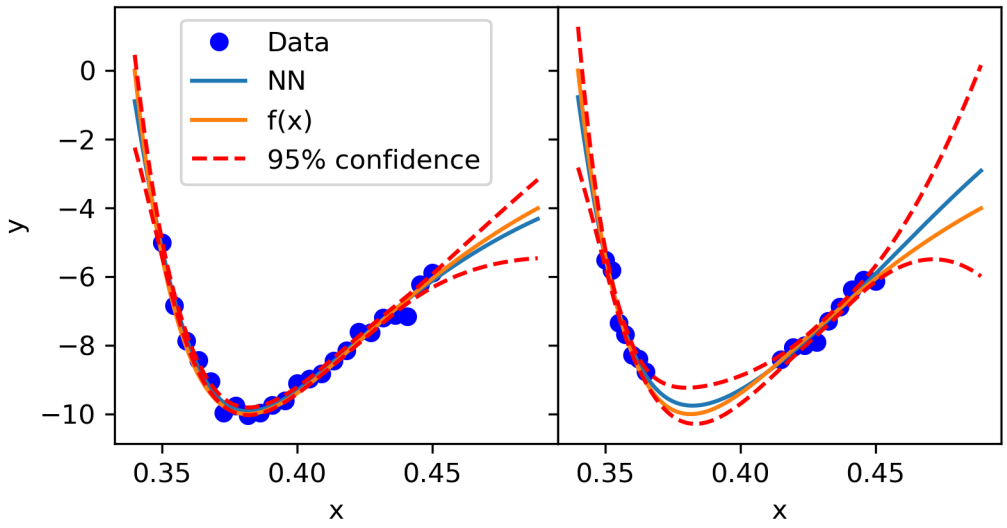
4.3 Results

We show examples of using the delta method on different models to demonstrate how the uncertainty behaves. We begin with a simple 1-D NN, and build complexity in subsequent examples.

4.3.1 One dimension input NN

This example is a one dimension input NN. We start with one dimension input for clearer intuition and visualization. We generated synthetic data from the one dimensional Lennard-Jones (LJ) function and added some Gaussian noise. We fitted this data to a neural network with structure [1, 4, 1] (one input, one hidden layer with four nodes, and one output) using `scipy.optimize.minimize`. The NN had 13 total parameters.

We test how the standard error changes with different training datasets. We generated two sets of training data to fit the NN, and Fig. 4.2 shows the fits. These sets of training data were from the same LJ function and had the same variance of Gaussian noise added. We expect the true function to be within the confidence interval 95% of the time. In Fig. 4.2a, the uncertainty increases for large and small x, which is desirable because we do not know how the NN will behave in those regions outside the training data. In Fig. 4.2b, there is a region of missing data in the middle, and the confidence interval expands in the region of missing data. These cases demonstrate that the uncertainty depends on the training data in a useful way. The uncertainty generally increases in regions with less data, which makes sense because we are less certain of our model in a space with less training data.



(a) (b)

Figure 4.2: One dimension input NN and confidence intervals. a): 23 training data points, and confidence interval wider at the edges. b): Region of missing data in middle, and confidence interval expands in region of missing data.

4.3.2 High dimensional NN potential

Trained NN potential

This example applies the delta uncertainty method to a high dimensional NN potential. We use the SingleNN (implemented in `PyTorch`) and weighted Behler-Parrinello style symmetry functions.[167–169] The data are DFT energy and force calculations based on atomic configurations, specifically the dataset used in Boes 2017.[170] The dataset contains 3,907 unique AuPd slabs, and example configurations are shown in Fig. 4.3. The symmetry functions transform the atomic configuration information into a vector of numbers, or "fingerprint", and we use four weighted G_2 symmetry functions. For the NN, we use two hidden layers with 11 nodes each; thus the NN architecture is [4, 11, 11, 2], which is 211 total parameters.

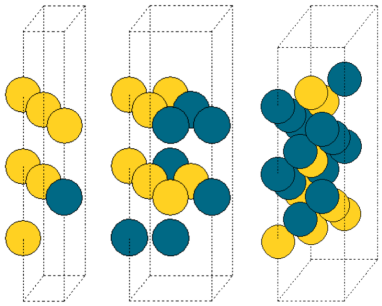


Figure 4.3: Three example atom configurations from dataset.

To demonstrate the usefulness of the uncertainty method, we start by training on a subset of the data. This mimics the iterative approach often used in training these models. We then check for extrapolation on the remaining data using the delta method. For this first potential, 572 configurations with a 3.934 Å lattice constant were randomly split into 64%, 16%, 20% train, validation, test sets, respectively. The NN was trained on energies and atomic forces using SingleNN, and uncertainties were calculated in the same `PyTorch` framework.

Fig. 4.4 shows the energy parity plots of the training, validation, and test sets. The parity is good in all cases, and root mean squared errors (RMSEs) are 0.003, 0.0023, 0.003 eV/atom for train, validation, and test, respectively. Fig. 4.5 shows the distributions of standard errors of confidence for the three datasets. The distributions are very similar and mostly overlapping. Fig. 4.6 shows the parity plot of the test dataset with 95% prediction intervals. The true values are within the prediction intervals for 98% of the dataset, which is close to 95% and shows the delta method provides quantitatively reasonable uncertainties in this case. The main result is that similar datasets with the same accuracy using the model will have similar distributions of uncertainties.

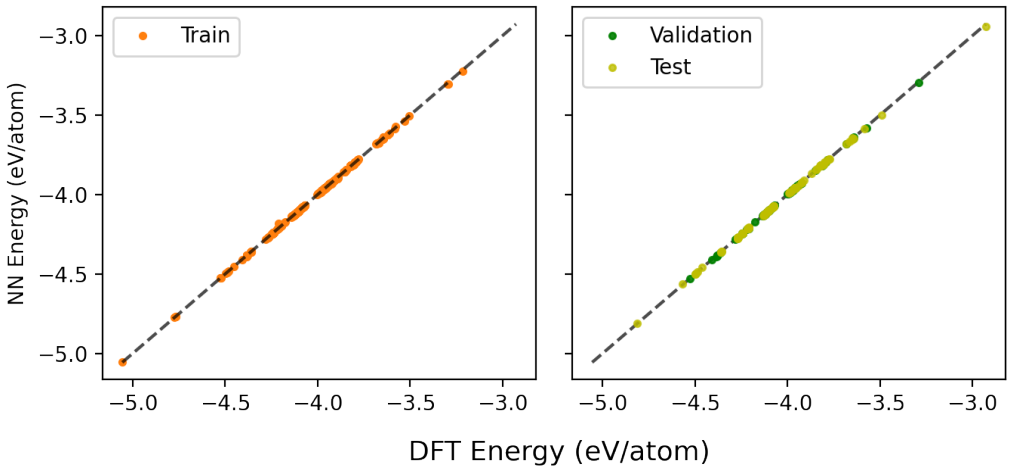


Figure 4.4: Parity plot of SingleNN.

Figure 4.5: Distribution of uncertainties (standard error confidence).

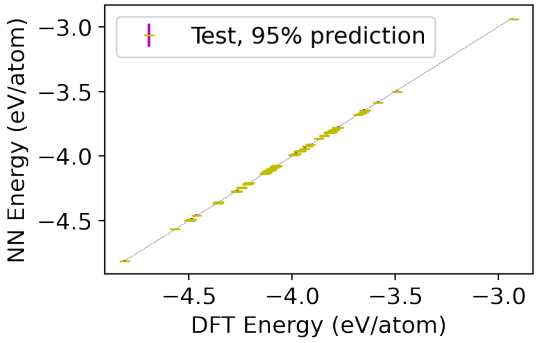


Figure 4.6: Parity plot with 95% prediction intervals for test set.

Next we use the same potential to predict on a new dataset. If the new dataset is dissimilar from the training data, we expect the uncertainties to be high. While the training set had 3.934 Å lattice constants, the new dataset has 4.034 and 4.134 Å lattice constants, which we will refer to as predict-4.0 and 4.1 datasets. As a result, we expect the fingerprints to differ from those of the train set, i.e. we know we are extrapolating here. Fig. 4.7 shows the energy parity plots for the predict sets with 95% prediction intervals. The predictions are offset with an error, and the uncertainties are clearly much larger than those for the test dataset from Fig. 4.6. Table 4.1 shows the average standard error of confidence/prediction for the datasets. Training and test datasets have around the same standard error confidence of 0.002 eV/atom, and predict-4.0 and 4.1 sets have higher uncertainties of 0.023 and 0.034 eV/atom, respectively, which are one order of magnitude larger than training and test. Since this uncertainty is much larger, it could indicate that the model is extrapolating on the predict datasets, and the parity plots (Fig. 4.7) seem to indicate this.

We examine the fingerprints, and Fig. 4.8 shows an example fingerprint for the train and predict datasets. There are regions where the predict-4.0 and 4.1 atoms' fingerprints are outside of the training distributions, which is suggestive of extrapolation. For predict-4.0, the true values are within the prediction intervals for 75% of the dataset, which is not that close to 95%, and for predict-4.1, the true values are within the prediction intervals for 0% of the dataset. This seems to indicate that the prediction interval becomes less quantitatively accurate as the extrapolation increases. However when the uncertainty is much larger than the training uncertainties, the model is likely extrapolating, and we should not trust the prediction. Therefore this uncertainty method helps identify the data regions where a model extrapolates. Fig. 4.9 shows the

standard error confidence vs. absolute energy error, and their distributions for test and predict datasets. Fig. 4.9 shows the general trend that uncertainty from the delta method increases when true error increases. The trend is most obvious in a heterogeneous dataset.

Table 4.1: Average standard errors of datasets

Dataset	Average Standard Error Confidence (eV/atom)	Average Standard Error Prediction (eV/atom)
Test	0.0020	0.0036
Predict 4.0	0.0234	0.0235
Predict 4.1	0.0336	0.0337

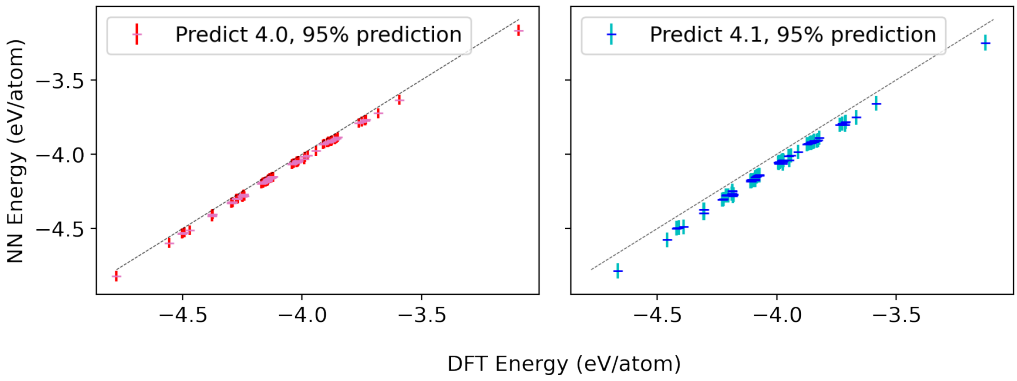


Figure 4.7: Prediction on new lattice datasets, uncertainty may be much larger in an extrapolation region.

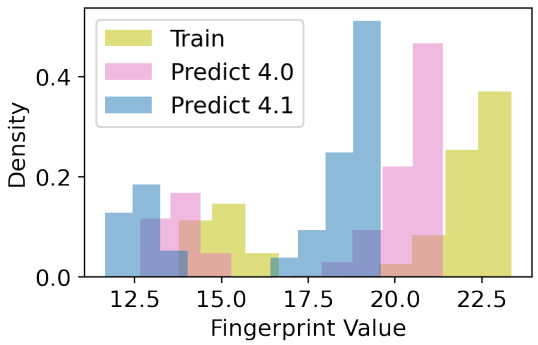


Figure 4.8: The predict-4.0 and 4.1 datasets have fingerprints outside of range of training distribution (fingerprint example shown is $\eta = 0$ with Pd center atoms).

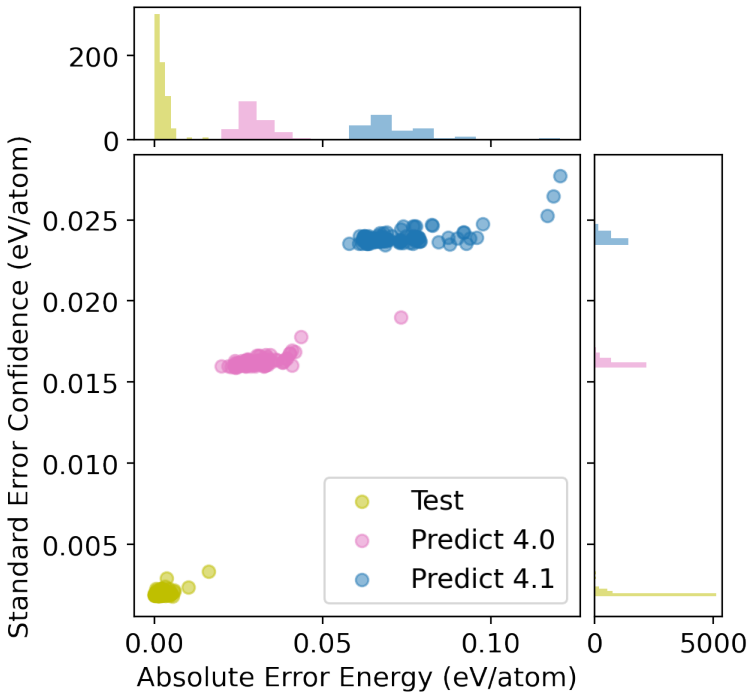


Figure 4.9: Standard error from delta method vs. absolute error and their distributions.

Uncertainties after retraining

Next we retrain the potential with some of the predict-4.0 and 4.1 data and check how uncertainties are affected. We expect the uncertainties to decrease after retraining. We added 64% of each predict-4.0 and 4.1 dataset, or 365 data points each, and retrained. Fig. 4.10 shows the energy parity plots of the new training and predict sets. After retraining, the predict set is on parity and no longer offset. The true values are within the prediction intervals for 98.7% of the training data and 98.5% of the predict data, which are close to the theoretical 95% and show the uncertainties calculated from the delta method are quantitatively reasonable. Fig. 4.11 shows the updated standard error confidence vs. absolute energy error, and their distributions for test and predict datasets. After retraining, the standard errors across datasets are mostly overlapping, and the average standard errors are the

same for the datasets. The average standard error confidence and predict are 0.002 and 0.003 eV/atom, respectively. Since we retrained on the predict-4.0 and 4.1 datasets, we are no longer extrapolating on that data and the uncertainties updated to reflect this: they are no longer an order of magnitude larger than the train sets' as was the case before retraining. We can use this uncertainty method to iteratively retrain a potential by adding data with high uncertainties. This is sometimes called active learning.

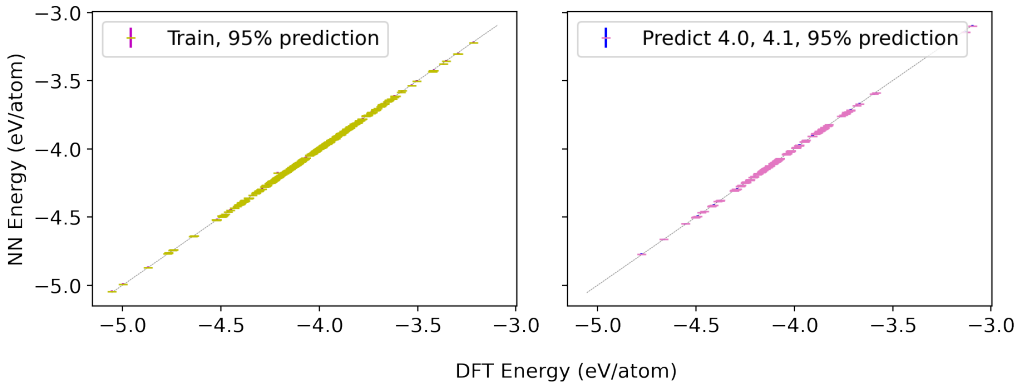


Figure 4.10: Parity plot after retraining.

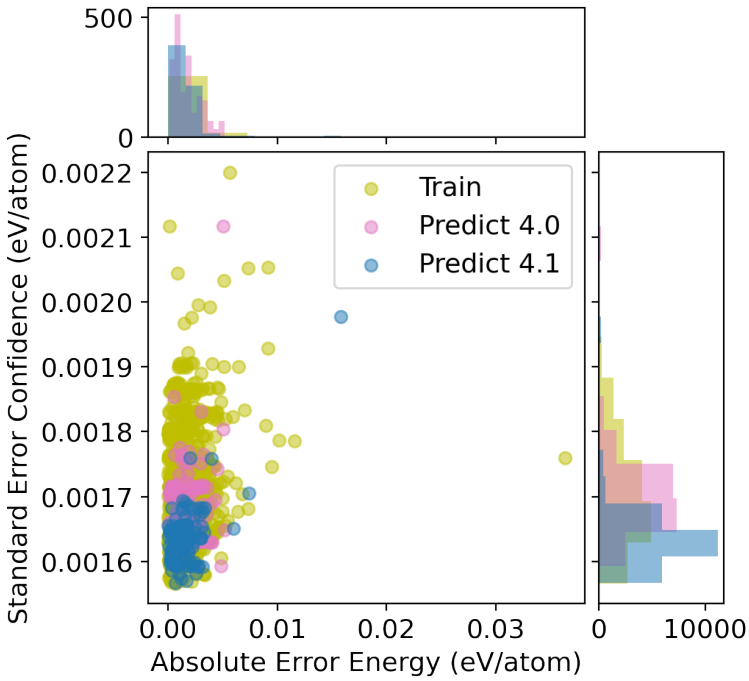


Figure 4.11: Distribution of uncertainties after retraining.

In the calculation of the Fisher information matrix, we used the errors of energies only, although we trained on energies and forces. From a theoretical perspective, the Fisher information should include some information about force errors, but exactly how much to include is nonobvious. By using only loss of energies, we save computational time for calculating the Fisher information, and the uncertainty measurement still accomplishes the objective and is quantitatively reasonable. Therefore in practice, using only the loss of energies for the Fisher information works well.

We can also extend uncertainty to other properties such as forces. For this case, in Equation 4.1, $g(\hat{\theta})$ is force, which is $-\frac{\partial E}{\partial \text{position}}$, where E represents energy. We obtain $g'(\hat{\theta})$ through automatic differentiation by taking the derivative of $-\frac{\partial E}{\partial \text{position}}$ with respect to model parameters. In this way, we can use the delta method to calculate uncertainties for other

quantities of interest. Further work can be done to investigate the quality and methods for force uncertainties of NN potentials.

There is a possibility for fast approximations of the Fisher information after retraining. If we retrain by adding one or a few new training points, we may want a cheaper calculation of the Fisher information matrix. The Fisher information matrix can be linearly separated by training data since the loss is a sum over training data points. If the parameters of the model did not change from retraining, then the new Fisher information is the summation of the original Fisher information and the Fisher information for the new training points. Since retraining likely alters the model parameters, the previous Fisher information from old model parameters is an approximation. If only a few training points are added and model parameters do not change much, taking the Fisher information of the new training points and adding it to the original can be a fast approximation of the true Fisher information. Further work is required to determine when this approximation is adequate.

4.4 Conclusions

The delta method is a fast and easy way to estimate uncertainty. It requires the Hessian of the loss and gradient with respect to model parameters, and these are obtainable with most machine learning packages using automatic differentiation. The delta method is applicable to most models that are parametric and have nonzero gradients with respect to parameters. The uncertainty estimate will depend on the training data, model, and input (point) for which the uncertainty is calculated. The delta method is an alternative to ensemble or bootstrapping methods for obtaining uncertainty estimates, and uncertainty estimates are important because they can help determine when a model is extrapolating and increase model reliability.

We showed an application of the delta method to a high dimensional NN potential in molecular simulation. We illustrated how we can iteratively retrain a model by adding data with high uncertainties to improve it. This could also be done on-the-fly, e.g., while running an MD simulation with an ML potential. The uncertainty can determine the longest timescale MD simulation that is valid for a potential, or to identify when additional data should be added to the training data to improve it. The utility of the delta method shown here extends far beyond molecular simulation, and it can also be applied to many other applications that rely on linear or nonlinear regression models.

5 Model Specific to Model General Uncertainty for Physical Properties

5.1 Introduction

Many physical properties are derived from models. For example, reaction rates in a chemical reaction network can be determined from a kinetics model.[171–173] Thermal diffusivity of a material can be derived from a model that includes the differential equations governing heat transfer.[174] We calculate properties such as diffusion, viscosity, density, elastic modulus from MD simulations and further modeling such as Arrhenius equation, adjustments for finite size effect, extrapolating to alternate conditions.[175–177] We would like to report the physical property and its sensitivity to design choices in model and data selection. We can classify uncertainty into its sources, and different frameworks have been proposed.[178] For deriving physical properties, the sources of uncertainty may have a clear interpretation, such as different sources of experimental error, model selection, and parameter uncertainty.[179] Another common framework describes uncertainty as aleatoric or epistemic. Aleatoric refers to randomness in the experiment, or observations, while epistemic refers to ignorance about the best model.[180,181] Based on this description, uncertainty from process variation and noise in observations can be considered aleatoric, while uncertainty in parameters, model specification, and dataset shift can be classified as epistemic uncertainty. Sometimes the different uncertainties are not clear cut[180] or not feasible to separate.[178]

The uncertainty could be in the form of a confidence interval (frequentist) or even a full probability distribution (Bayesian). In Chapter 4, we described

some methods of obtaining uncertainty including ensembles, delta method, and Gaussian process, while focusing on the delta method. Apart from Gaussian process, the methods were specific to standard nonlinear regression. A point estimate of the parameters was obtained by minimizing mean squared error, and the parameter and model prediction standard errors were obtained from the separate methods described. Determining a point estimate of the parameters and the confidence intervals falls under the frequentist approach. In an alternate setting, one could update a distribution over parameters as data is observed, and obtain uncertainties using the distribution. This would be a Bayesian approach, and examples of methods under this setting include probabilistic graphical models, Bayesian regression, and Gaussian process.[182]

Given the various possible sources of uncertainty and methods of calculation, this chapter aims to answer "what is the uncertainty really telling us". We use an example of obtaining physical properties from an equation of state for solid. The physical properties are equilibrium volume, energy, and bulk modulus. We obtain uncertainties using three methods: delta method, Bayesian nonlinear regression, and Gaussian process. Through comparison between uncertainties obtained from the methods, we show that the delta method and Bayesian nonlinear regression give model specific uncertainty while Gaussian process gives model general uncertainty, although they can all be considered epistemic uncertainty. In Section 5.1.1, we provide background and motivation for the equation of state problem. In Section 5.1.2, we provide brief description of probabilistic modeling and further applications in engineering.

5.1.1 Equation of state

The equation of state (EOS) of a solid relates pressure-volume or energy-volume. The EOS is important in fields such as materials, condensed matter, and geophysics, and can be used to extract the equilibrium volume and bulk modulus.[183] The pressure P, bulk modulus B can be expressed as:

$$P = -\frac{dE}{dV} \tag{5.1}$$

$$B = -V\frac{dP}{dV} = V\frac{d^2E}{dV^2} \tag{5.2}$$

where E is energy and V is volume. From Eq. 5.2, in an energy-volume EOS, the bulk modulus depends on the second derivative, thereby increasing its variance. Many physical properties of interest rely on derivatives, have higher variance, making it more important to quantify their uncertainty.

Researchers have developed many EOS of different analytical forms, and they aim to describe a range of solids accurately.[184–188] Typically the analytical function has some parameters which are fit to experimental or computational data using nonlinear regression. It is not necessarily clear which analytical function is most accurate for obtaining physical properties of a particular solid. Therefore uncertainty exists with regard to model selection and a specific model's parameters.

Many of the analytical functions have the equilibrium energy, volume, and bulk modulus as parameters themselves, which makes their standard error available from the inverse Fisher information matrix (Section 4.2). For other functions, it is possible to calculate the uncertainty of the physical property under nonlinear regression and the delta method when the derivative of the property w.r.t. function parameters exists.

5.1.2 Probabilistic models in engineering applications

A probabilistic graphical model (PGM) uses a graph to represent a joint probability over variables. The graph encodes a set of conditional independence assumptions between its nodes, which represent the random variables. A probability distribution that satisfies the independencies associated with the graph can be factorized according to the graph, which is a theorem.[189] This is helpful because it is often more intuitive to first specify a graph over variables and then derive a factorized joint distribution. Some examples of PGMs are Hidden Markov models, Naive Bayes, and Bayesian regression. Fig. 5.1 shows a graphical representation of Bayesian regression. An equivalent interpretation is the joint probability factorized according to the chain rule and conditional probabilities of the graph, shown in Eq. 5.3.

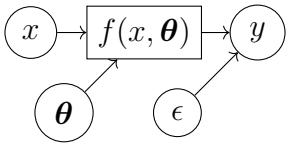


Figure 5.1: Graphical representation of Bayesian regression.

$$p(x, y, \boldsymbol{\theta}, \epsilon) = p(x)p(\boldsymbol{\theta})p(\epsilon)p(y|x, \boldsymbol{\theta}, \epsilon) \tag{5.3}$$

A PGM can be advantageous to use when some random variables have a clear natural distribution, such as Poisson, Bernoulli, or when variables have clear dependence or independence relationships. Domain knowledge can identify high relevance variables and reasonable independence relations to build a PGM that closely represents reality.

Many engineering and science applications can benefit from PGMs, and we review some here. One benefit of probabilistic models is further customization of error structure among observations, as shown by Miki et al.

to deduce ionization rate of atomic nitrogen.[171] Napp et al. used PGM to model chemical reaction networks.[173] Other physical properties such as modulus of elasticity[190] and crack propagation in materials[191] were found using probabilistic models. Researchers have integrated chemical process knowledge with probabilistic modeling. Bayesian regression was used to model pressure for safe and reliable operation in a natural gas regulating and metering station.[192] Lu et al. used time-series PGM for regression of variables in a hydrocracking process,[193] and Chen et al. used probabilistic principal components analysis and mixture model for trend and fault identification of industrial propylene polymerization.[194] There are clearly many applications of PGMs and Bayesian methods, also in the pharmaceutical industry[195] and astronomy field.[196]

Another useful probabilistic model is the Gaussian process. A Gaussian process (GP) is a collection of random variables such that any finite number of them have a joint multivariate normal distribution. The distribution of the GP is a distribution of functions with a continuous domain. For the EOS example, we develop a GP with joint distribution over a function and its first and second derivative observations to easily calculate bulk modulus and minimum volume. We describe our implementation in Section 5.2.2. Including derivatives in the collection of GP random variables can be useful to train with function observations and derivatives, such as energy and forces. Indeed, the Gaussian Approximation Potential (GAP) uses GP to fit energies, forces, and virial stress (derivative of energy w.r.t. lattice deformation) of atomic configurations.[47,197–203] GP have also been used as surrogate models to speed up nudged elastic band calculation or geometry optimization, sometimes using their uncertainty to guide training dataset selection.[204–206] GP model was used to predict yield and lower heating value of fluidized bed gasifier.[207]

5.2 Methods

5.2.1 Approximate inference for PGMs

Given a PGM, we typically want to answer queries about its probability distribution. This is called inference. One of the most common queries is computing the posterior distribution, or conditional probability of random variables in the PGM given data. In the Bayesian regression in Section 5.1.2, the posterior distribution is $p(\boldsymbol{\theta}, \epsilon|x, y)$. To compute the distribution, we use approximate inference methods, meaning they approximate the target distribution. Specifically we use variational inference and Markov chain Monte Carlo (MCMC). We focus on these methods because they are widely used for many Bayesian models. Variational inference is usually faster than MCMC, while MCMC is guaranteed to asymptotically produce samples from the true target distribution.[208,209]

Variational inference

In variational inference, we approximate the posterior using a surrogate distribution called the variational distribution q. Continuing the Bayesian regression example, our variational distribution is $q(\boldsymbol{\theta}, \epsilon)$. We assume a form for q, and some common choices are multivariate normal or mean-field, which is a product of independent normals for each parameter. To find q, we minimize the KL-divergence between q and the true posterior, $KL(q(\boldsymbol{\theta}, \epsilon)||p(\boldsymbol{\theta}, \epsilon|x, y))$. The KL has an intractable term $\log p(x, y)$, so we remove it from our optimization, and actually minimize $\mathbb{E}_q[\log q(\boldsymbol{\theta}, \epsilon) - \log p(\boldsymbol{\theta}, x, y, \epsilon)]$, which is the negative ELBO (evidence lower bound).[208] Minimizing KL is equivalent to maximizing ELBO.

Based on Eq. 5.3, we can factorize the joint probability, and our minimization problem is

$$q^*(\boldsymbol{\theta}, \epsilon) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}}\, \mathbb{E}_q[\log q(\boldsymbol{\theta}, \epsilon) - \log p(\boldsymbol{\theta}) - \log p(\epsilon) - \log p(y|\boldsymbol{\theta}, \epsilon, x) - \log p(x)] \tag{5.4}$$

We note the term $\log p(x)$ does not affect the optimization solution for q. Also, the terms on the right appear in maximum likelihood estimation (MLE) and maximum a posteriori (MAP) estimation. Specifically one could solve

$$\hat{\boldsymbol{\theta}}_{\text{MLE}}, \hat{\epsilon}_{\text{MLE}} = \underset{\boldsymbol{\theta}, \epsilon}{\operatorname{argmax}} \log p(y|\boldsymbol{\theta}, \epsilon, x) \tag{5.5}$$

$$\hat{\boldsymbol{\theta}}_{\text{MAP}}, \hat{\epsilon}_{\text{MAP}} = \underset{\boldsymbol{\theta}, \epsilon}{\operatorname{argmax}} \log p(\boldsymbol{\theta}) + \log p(\epsilon) + \log p(y|\boldsymbol{\theta}, \epsilon, x). \tag{5.6}$$

The difference here is that MLE and MAP return point estimates while the optimization of 5.4 returns a distribution q.

In our model, we must define prior probability distributions for $\boldsymbol{\theta}$ and ϵ. We also select a form for q, which was multivariate normal. Now we estimate the parameters of q, in our case mean and covariance matrix, by gradient descent and estimates of the ELBO gradient. This method is stochastic variational inference (SVI),[210–212] and software such as `pyro` and `tensorflow-probability` have implemented SVI and automatic calculation of ELBO and its gradients.

We used `pyro` and defined normal distribution priors for parameters in the EOS models. The prior for ϵ was normal with mean zero and standard deviation with prior Uniform$(0, 1)$. Negative ELBO was minimized with Adam optimizer[213] until it stopped decreasing.

Markov chain Monte Carlo

Markov chain Monte Carlo methods generate samples from a probability distribution. A proposed sample is accepted or rejected according to some criterion related to the target distribution. The next sample is generated by random walk or another algorithm. Hamiltonian Monte Carlo (HMC) is an efficient and well-studied MCMC method, and uses Hamiltonian dynamics from physics to propose the next sample.[214] We used the NUTS method[215] to generate HMC samples of posterior, implemented in `pyro`, and inspected the sample trajectory to retain samples after convergence.

5.2.2 Gaussian process joint including derivatives

A function f is a Gaussian process with mean $m(x)$ and covariance kernel $k(x_i, x_j)$ if

$$\begin{aligned} [f(x_1), ..., f(x_n)] &\sim \mathcal{N}(\mu, K) \\ \mu_i &= m(x_i) \\ K_{ij} &= k(x_i, x_j). \end{aligned} \tag{5.7}$$

To perform inference with a GP (predict on test data), we note that the joint distribution between noisy training data $\mathbf{y} = f(X) + \epsilon$ and test data $f(X_*) = \mathbf{f}_*$, where $\epsilon \sim N(0, \sigma^2)$, is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{5.8}$$

GP is assumed to have zero mean for convenience. To deal with nonzero means, we can subtract $m(X)$ to create a zero mean GP and add back $m(X_*)$ after inference.

We use the conditional rule for multivariate normals, therefore $p(\mathbf{f}_*|X, \mathbf{y}, X_*)$ is multivariate normal with mean $\bar{\mathbf{f}}_*$ and $\text{cov}(\mathbf{f}_*)$, where

$$\bar{\mathbf{f}}_* = K(X_*, X)[K(X, X) + \sigma^2 I]^{-1}\mathbf{y} \tag{5.9}$$
$$\text{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2 I]^{-1}K(X, X_*).$$

Now suppose $\mathbf{y}$ and $\mathbf{f}_*$ may include arbitrary number and location of derivative and second derivative observations, for example

$$X = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} f(X_0) \\ f'(X_1) \\ f''(X_2) \end{bmatrix}. \tag{5.10}$$

Then the covariance matrices $K(X, X), K(X, X_*), K(X_*, X), K(X_*, X_*)$ must include covariances between functions and derivatives, functions and second derivatives, derivatives and derivatives, etc. The covariances are

$$\text{cov}\left(f(x_i), \frac{\partial f(x_j)}{\partial x_j}\right) = \frac{\partial k(x_i, x_j)}{\partial x_j}, \quad \text{cov}\left(\frac{\partial f(x_i)}{\partial x_i}, \frac{\partial f(x_j)}{\partial x_j}\right) = \frac{\partial^2 k(x_i, x_j)}{\partial x_i \partial x_j} \tag{5.11}$$

etc.[131] Suppose $X_0 \in \mathbb{R}^{n\times 1}, X_1 \in \mathbb{R}^{m\times 1}$, we define an "element-wise" derivative

$$\frac{\partial K(X_0, X_1)}{\partial_\odot X_1^T} = \left[\frac{\partial K(X_0, x_{1,1})}{\partial x_{1,1}}, ..., \frac{\partial K(X_0, x_{1,m})}{\partial x_{1,m}}\right] \in \mathbb{R}^{n\times m}. \tag{5.12}$$

Therefore $K(X,X), K(X,X_*)$, etc. have blockwise covariances,[216] as an example

$$K(X,X) = \begin{bmatrix} K(X_0,X_0) & \frac{\partial K(X_0,X_1)}{\partial_\odot X_1^T} & \frac{\partial^2 K(X_0,X_2)}{(\partial_\odot X_2^T)^2} \\ \frac{\partial K(X_1,X_0)}{\partial_\odot X_1} & \frac{\partial^2 K(X_1,X_1)}{\partial_\odot X_1 \partial_\odot X_1^T} & \frac{\partial^3 K(X_1,X_2)}{\partial_\odot X_1 (\partial_\odot X_2^T)^2} \\ \frac{\partial^2 K(X_2,X_0)}{(\partial_\odot X_2)^2} & \frac{\partial^3 K(X_2,X_1)}{(\partial_\odot X_2)^2 \partial_\odot X_1^T} & \frac{\partial^4 K(X_2,X_2)}{(\partial_\odot X_2)^2 (\partial_\odot X_2^T)^2} \end{bmatrix}, \tag{5.13}$$

and we apply Eq. 5.9 as before.

In practice, the kernel function k is selected, and its parameters are optimized. Some kernel functions are linear, periodic, and radial basis function (RBF), and selecting a kernel is described in Duvenaud 2014.[217] We used the RBF kernel, Eq. 5.14, because it works well for many functions.

$$k(x_i, x_j) = \lambda \exp\left(-\frac{(x_i - x_j)^2}{2l^2}\right) \tag{5.14}$$

The RBF kernel has two parameters, scale λ and lengthscale l. We optimized these parameters using `gpytorch` and by maximizing the log marginal likelihood $p(\mathbf{y}|X)$.[131] We used vectorized-map and automatic differentiation in `jax` to efficiently calculate derivatives as those required in Eq. 5.13.

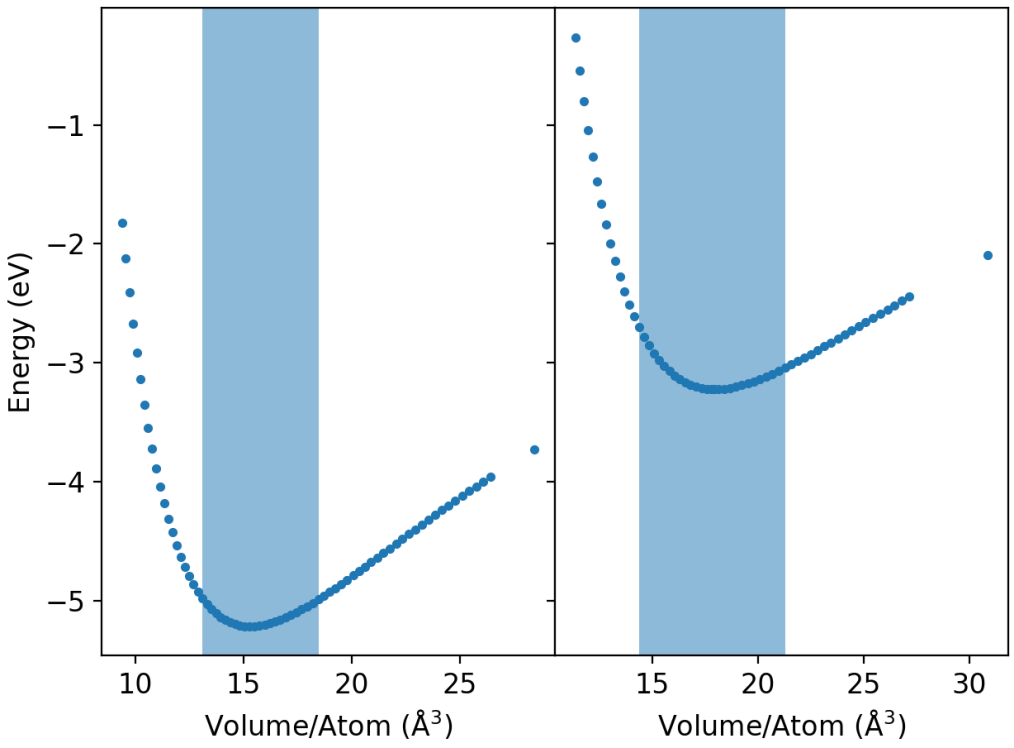
To find the posterior over minimum volume, energy, and bulk modulus, we took 1,000 joint samples of the function, first and second derivatives at 1,000 linspaced points across the volume range and smoothed them with `interp1d`. To find minimum volume, we used the root-finder `brentq` in `scipy` over the first derivative.[218] The corresponding sampled energies at the volumes were the distribution of minimum energies, and corresponding second derivatives multiplied with minimum volumes were the distribution of bulk moduli.

5.3 Results

We discuss the data for the EOS and show results for uncertainties from nonlinear regression and the delta method, Bayesian regression, and the Gaussian process.

5.3.1 Data

Fig. 5.2 shows the data for FCC Pd and Au using DFT calculation. The data are from Boes 2016 and 2017.[35,219] The EOS calculated in the papers used all of the data points shown in Fig. 5.2. We used the data closer to the minimum, the shaded regions: 24 and 29 points for Pd and Au, respectively. We chose the narrower range to maintain homoscedastic residuals across the different EOS models. The range of data for training makes a significant difference in the physical properties, and the spread across models. In addition, there is not a clear way to determine which range of data should be used. This is further motivation for reporting uncertainties, and the uncertainties reported depend on the data selected.



(a) Pd (b) Au

Figure 5.2: Data for EOS. Shaded region represents data used in training.

5.3.2 Nonlinear regression

We perform nonlinear regression for common equations of state, and Table 5.1 shows the result. The error is low for all of the models. Most minimum volumes fall within 0.01 and 0.03 Å^3 for Pd and Au, respectively. The minimum energies are almost the same for all models, and the bulk moduli fall within 15 and 20 GPa for Pd and Au, respectively. If we used the entire dataset from Fig. 5.2, the ranges for minimum energies, volumes, and bulk moduli would be much wider, and the point estimates would be different. For example, using SJ and the entire datasets, the bulk moduli for Pd and Au are both 6 GPa larger than our results.[35,219]

Table 5.1: Equilibrium volume, equilibrium energy, bulk modulus from different equations of state

	Equation of State	V ($Å^3$)	E (eV)	B (GPa)	RMSE (eV)	MAE (eV)
Pd	Stabilized Jellium[183]	15.304	-5.215	169	1.0e-4	8.5e-5
	Anton-Schmidt[220]	15.303	-5.214	168	7.4e-5	6.6e-5
	Polynomial3/Taylor	15.331	-5.216	180	1.9e-3	1.7e-3
	Murnaghan[221]	15.302	-5.214	165	4.6e-4	4.0e-4
	Birch/Birch-Murnaghan[222]	15.303	-5.214	168	6.8e-5	6.1e-5
	Poirier-Tarantola[185]	15.307	-5.215	171	3.9e-4	3.3e-4
	Vinet[222]	15.303	-5.215	168	4.1e-5	3.3e-5
Au	Stabilized Jellium	17.960	-3.222	141	2.1e-4	1.7e-4
	Anton-Schmidt	17.968	-3.221	139	3.2e-4	2.6e-4
	Polynomial3/Taylor	17.927	-3.225	160	3.9e-3	3.4e-3
	Murnaghan	17.984	-3.221	137	9.5e-4	8.2e-4
	Birch/Birch-Murnaghan	17.967	-3.221	140	2.8e-4	2.2e-4
	Poirier-Tarantola	17.950	-3.222	144	7.4e-4	6.3e-4
	Vinet	17.963	-3.222	140	1.8e-4	9.6e-5

Next we calculated standard errors for a set of the models, shown in Table 5.2. We chose different models across Pd and Au to show generalizability of our results, and we chose models for each element that had slightly different bulk modulus. The delta method follows Sections 4.2 and 4.2.1. The standard errors are larger for models with higher RMSE because the inverse Fisher information is scaled with the model error. The standard errors confidence for minimum volume and energy are quantitatively reasonable given the spreads in Table 5.1. For bulk modulus, if we consider 95% confidence interval ($\pm \sim 2 \cdot$ s.e.), the selected models' intervals do not overlap with each other. Hence the uncertainties from the delta method are inherent to the model used, which we describe as model-specific uncertainty.

Table 5.2: Standard error confidence of physical properties from delta method

		Standard Error Confidence		
	Equation of State	V (Å^3)	E (eV)	B (GPa)
Pd	Birch/Birch-Murnaghan	0.0011	0.0001	0.13
	Poirier-Tarantola	0.0060	0.0004	0.82
Au	Murnaghan	0.0138	0.0010	1.40
	Vinet	0.0028	0.0003	0.25

5.3.3 Bayesian regression

We performed Bayesian regression on the same set of models. We used stochastic variational inference (SVI) with multivariate normal variational distribution and HMC to find the posterior distribution over parameters, as described in Section 5.2.1. For SVI, we used priors over parameters that were close to the nonlinear regression solution. This is analogous to choosing a good initial guess for nonlinear regression, and greatly helped the optimization. The ELBO estimates were noisy, and optimization for one model required around 5 minutes on a laptop. The HMC is more robust to the initial guess, however it required around 20 minutes for one model. Fig. 5.3 compares the posteriors from SVI, HMC, and the 95% confidence interval from the delta method for Pd Poirier-Tarantola. Figures for additional models are in Appendix E. In Bayesian setting, the credible interval is analogous to the prediction interval, and $1-\alpha$ credible interval for θ is defined as $\int_a^b p(\theta|x,y)d\theta = 1-\alpha$. The statistical interpretation is technically different than a confidence or prediction interval, but researchers have shown that they are quantitatively similar.[223–225] We note that Fig. 5.3 shows the marginal posteriors over parameters, although we did find the joint posteriors. The HMC posterior means are close to the parameters found using nonlinear regression. Between SVI and HMC, HMC predicts a tighter posterior with a mean closer to the

nonlinear regression result. This is expected because HMC can predict the posterior more accurately than variational inference. For this model, the delta method 95% confidence intervals are wider than the SVI or HMC credible intervals. However, they are still quantitatively close, which was also observed for the other models. This indicates that uncertainty from Bayesian regression in this setting is also model-specific.

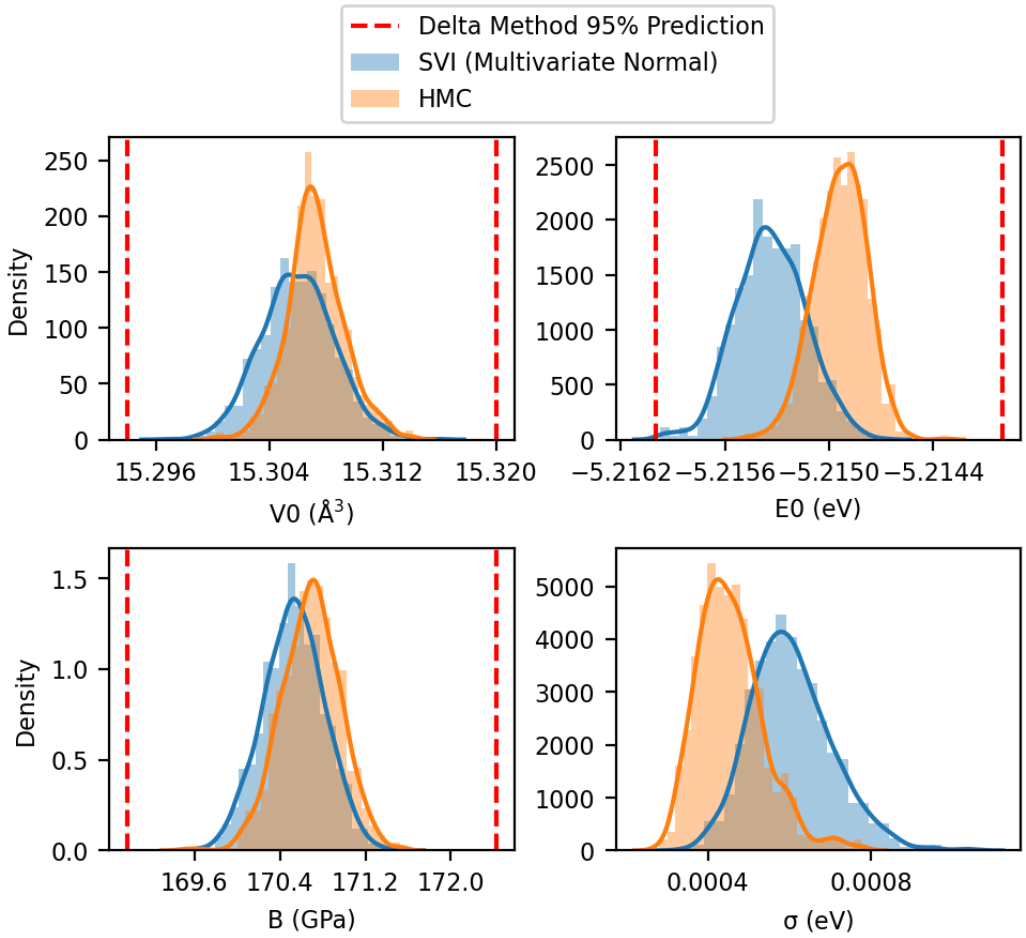


Figure 5.3: Comparison of HMC, Variational inference posteriors and delta method confidence interval for Pd Poirier-Tarantola.

5.3.4 Gaussian process

In this section, we show uncertainties from the Gaussian process for Pd. Results for Au are in Appendix E. Fig. 5.4 shows the Gaussian process

posterior. Each sample is from the joint distribution over the function, first and second derivatives. This means that for one sample, the function, first and second derivatives correspond with each other. The variance, especially around the lowest and highest volumes, grows larger as the derivative order increases. Clearly we would not expect the GP to extrapolate well outside of the training data range.

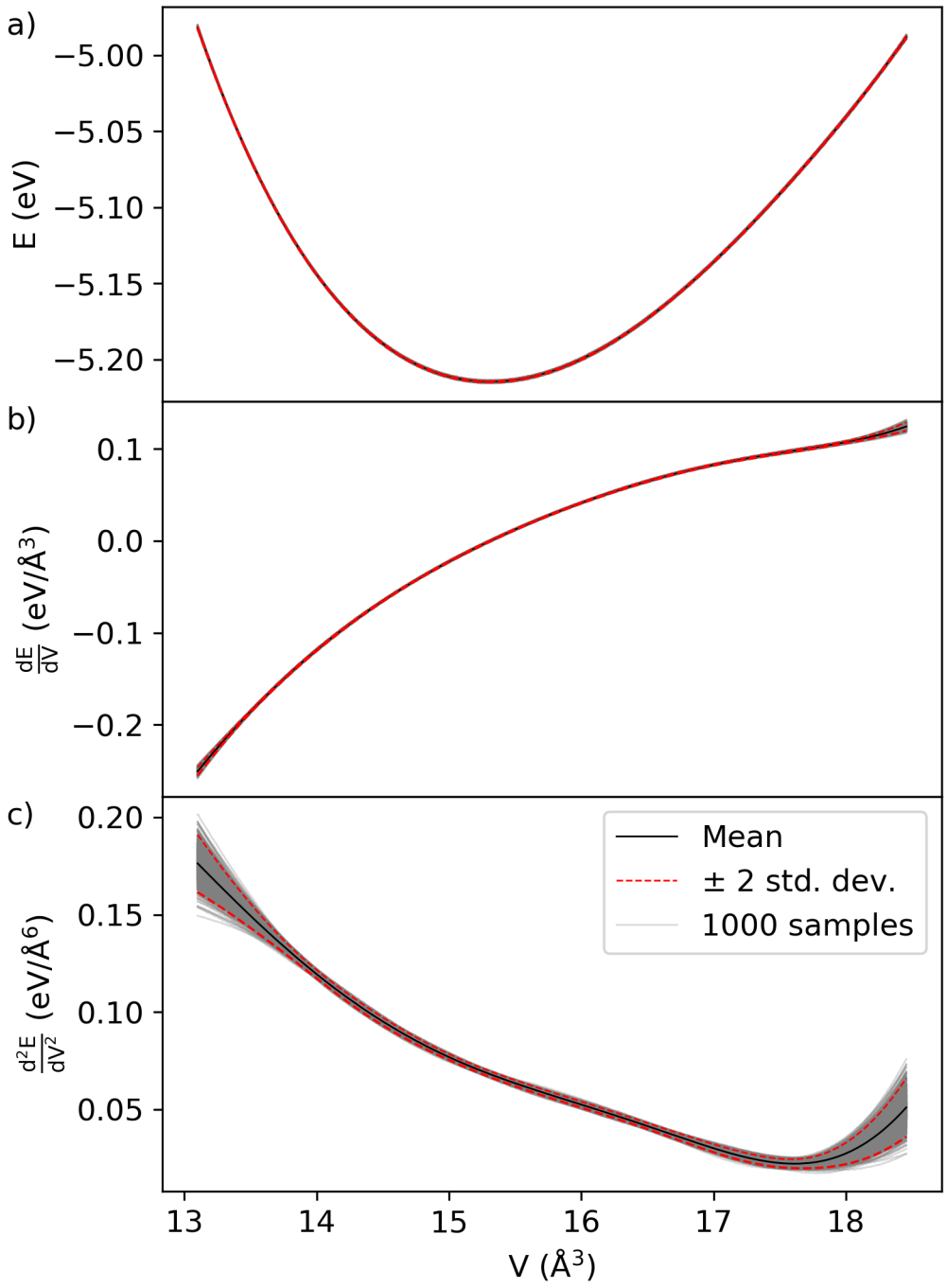


Figure 5.4: Gaussian process posterior for Pd EOS. a): function, b): first derivative, c): second derivative.

Figs. 5.5, 5.6, 5.7 show the Gaussian process distribution compared with previous methods for minimum volume, energy, and bulk modulus,

respectively. For all of the physical properties, the GP has the widest distribution compared with other methods. The GP distribution covers the predictions for almost all EOS models, while Bayesian regression posteriors do not always overlap across models, especially for minimum energy and bulk modulus, Figs. 5.6 and 5.7. The delta method intervals do not overlap across models for bulk modulus. The results show that GP uncertainties are model-general because their posterior distributions include almost any model that could be the true EOS.

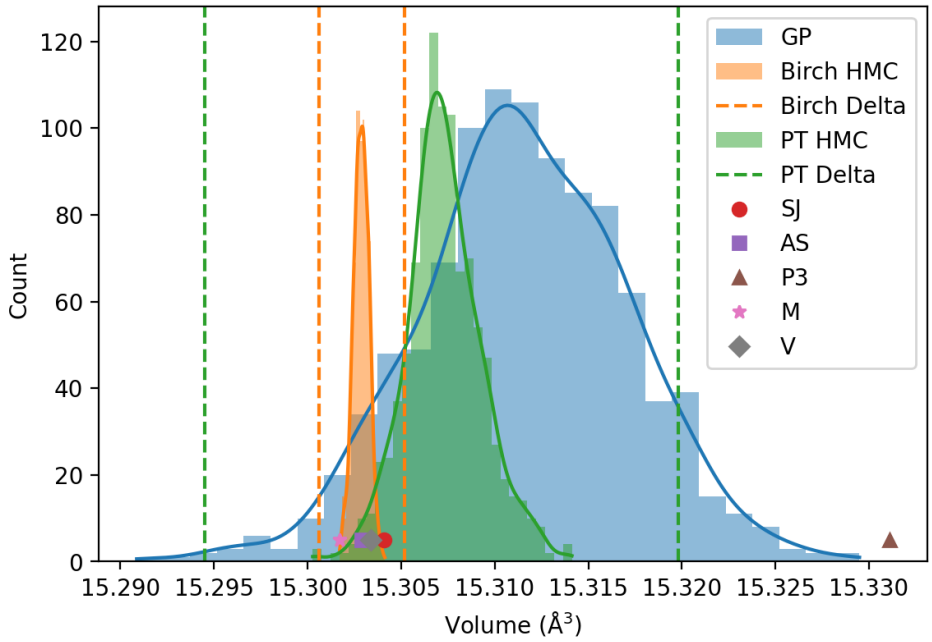


Figure 5.5: Comparison of GP, Bayesian regression, and nonlinear regression uncertainties with different model predictions for minimum volume.

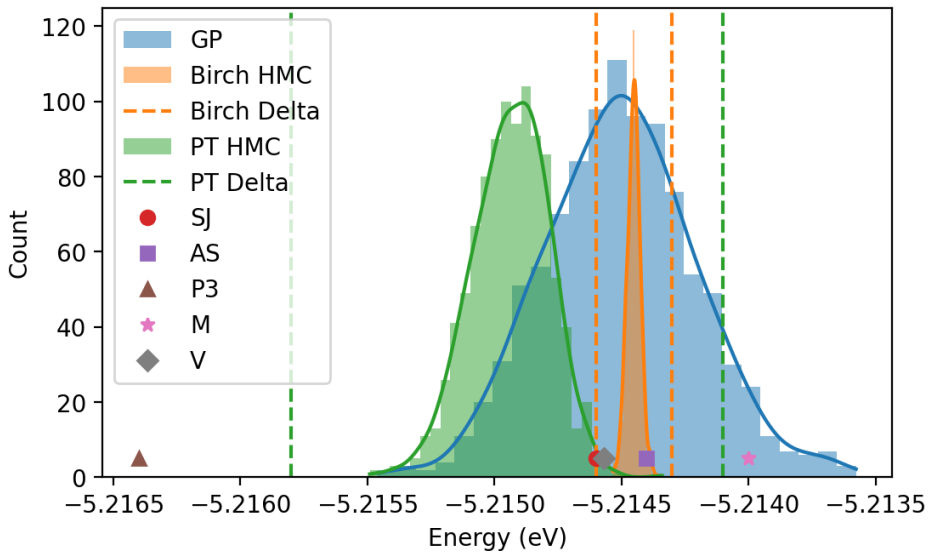


Figure 5.6: Comparison of GP, Bayesian regression, and nonlinear regression uncertainties with different model predictions for minimum energy.

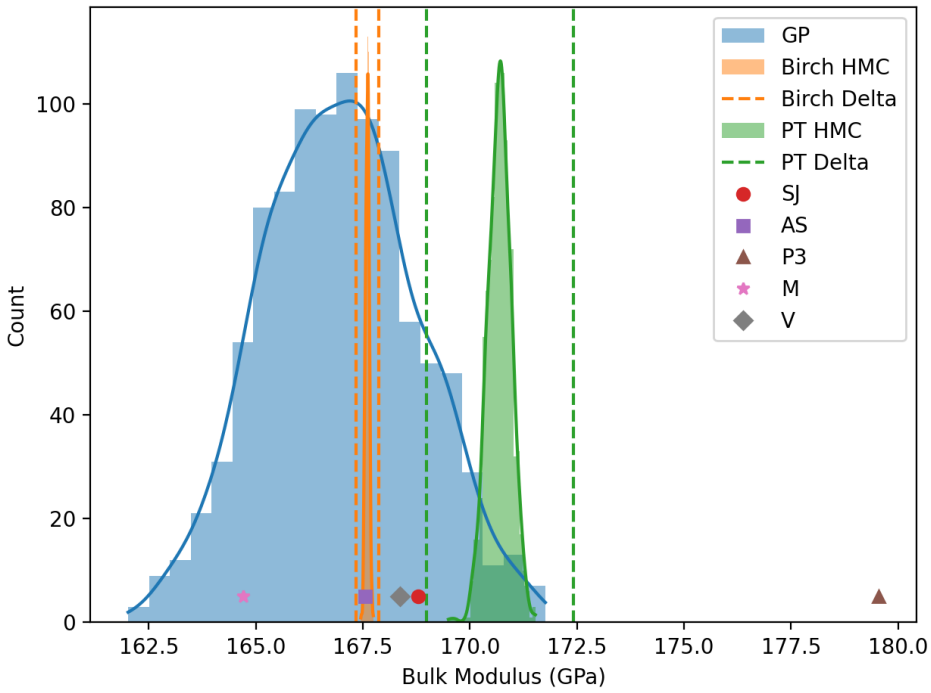


Figure 5.7: Comparison of GP, Bayesian regression, and nonlinear regression uncertainties with different model predictions for bulk modulus.

5.3.5 Overall comparison of methods

Here we compare the methods, including their ease of use. Between nonlinear and Bayesian regression, nonlinear regression is easier to set up, runs much faster, and is more commonly used. We showed that the delta method uncertainties are quantitatively close to those from Bayesian regression. Therefore if model parameters are expected to be normally distributed, nonlinear regression is recommended.

Bayesian regression is harder to set up. At the time of writing, `pyro` is generally easier to use than `tensorflow-probability`. For each problem, the probabilistic model and variational distribution must be defined. The software improve ease of use with automatically calculated ELBO and integrated optimizers. In these problems, Bayesian regression was slower than other methods, requiring around 5 and 20 minutes for SVI and HMC, respectively. Bayesian regression may outperform nonlinear regression if the parameters have a non-normal distribution, such as skewed, uniform, log-normal, discrete distribution, etc.

We showed both nonlinear and Bayesian regression give model-specific uncertainty. If uncertainty across models is required, the implementations would need to be modified and rerun for separate models. To avoid this, we can use Gaussian process. GPs are relatively easy to use with many available software such as `scikit-learn` and `gpytorch`. Optimizing the hyperparameters for our models required a few minutes, and calculating the covariance and distribution samples was even faster. However GPs naively scale with $O(n^3)$ with n as number of data points. There are ongoing research to improve GP scaling.[149,226–230] We showed that the GP uncertainty is larger than other methods and includes uncertainty from model selection. Using GP,

we can report uncertainty across a large model space without repeatedly fitting many different model forms.

5.4 Conclusions

Uncertainty arises from different sources, and there are alternate methods of calculating uncertainty. We reviewed background and literature applications in science and engineering of probabilistic graphical models and Gaussian process. There may be many analytical forms of a model that calculates physical properties, for example an equation of state. This further motivates an analysis of what sources of uncertainty are included in a particular uncertainty calculation. We compare nonlinear regression, Bayesian regression, and Gaussian process for their uncertainty quantification. Nonlinear regression gives a point estimate of the physical property, dependent on the specific model. The delta method can find a standard error and confidence or prediction interval in most cases. The delta method interval is wider for models with higher error, and is close to the posterior found from Bayesian regression. We used HMC and SVI for Bayesian inference, and HMC gave more consistent posteriors than SVI, at an expense of higher compute. Both the delta method and Bayesian regression give model-specific uncertainty, which is not expected to include uncertainty arising from model selection. We developed the joint GP over a function, its first and second derivatives to conveniently find distributions of physical properties requiring derivatives. The GP uncertainties cover almost all of the alternative EOS models' physical property predictions. Hence we conclude that GP gives model-general uncertainty which includes uncertainty from model selection.

6 Conclusions

This dissertation applied machine learning (ML) to address challenges in atomic simulation. ML allowed progress in analyzing structure of data, accelerating high accuracy computation, and quantifying uncertainty of models. In the future, we expect more integration between ML and simulation methods.

6.1 ML potentials to accelerate simulations

We built a neural net potential for a complex liquid alloy system. The Ni-Al-W system was especially challenging because it is liquid and has three chemical elements with sparse W composition. All of these reasons increased the system dimensionality. Ideally, we also wanted to investigate more temperatures, volumes, and compositions, which would further increase system complexity. We successfully integrated the potential with molecular dynamics simulation and obtained similar results to *ab initio* with NVT.

There are several key takeaways from the work. For parameterized fingerprints, the hyperparameter search must be completed every time the system changes chemically or structurally, as the hyperparameters can significantly change the potential accuracy. In future work, it would be best to automate the parameter search. Another takeaway is that if the MD runs for enough time steps, the configuration will likely reach an extrapolation region. Therefore we need to determine the longest MD simulation required for data collection and plan accordingly for the size and diversity of training dataset. We believe the delta method uncertainty could be integrated with MD to determine when the system reaches extrapolation. Currently, integrating ML models with high performance MD code is a challenge in the field. Researchers have developed `jax-MD` with differentiable MD and easy integration between

ML potentials and MD.[231] There could be promising future work with differentiable MD,[232] however the code is currently not as fast as production MD code such as `LAMMPS`.

We used the Behler-Parrinello NN, however the number of G^3 fingerprints (and therefore model complexity) scales quadratically with the number of chemical species.[169] In addition, many models with graph and convolution-based fingerprints are reaching state-of-the-art performance.[233–243] In future work, these models could be tested using the liquid alloy system. For the best model, extrapolation in MD may still be a concern, and the solution may be checking uncertainty during MD steps and reverting to physical potential (or DFT) for high uncertainty and the ML potential otherwise.

6.2 Extensions in analyzing liquid atomic configurations

We used MD simulation with a physical potential for liquid Al-Si. The diffusion and viscosities were calculated at many temperatures, and a Stokes-Einstein deviation occurred at temperatures near the melting point. We found significant clusters using Voronoi tessellation and agglomerative clustering, and developed per-atom diffusion and viscosity methods. The clusters were found to have a minimal impact on diffusion but a measurable impact on viscosity, which could explain the observed Stokes-Einstein deviation.

One of the challenges as simulations move toward longer length scales is increased data generation. We are already saving thermodynamic data and especially atomic data at limited frequency to save on storage space. Large amounts of data require longer times and are harder to analyze and transfer. Researchers have set up distributed analysis systems to combat these issues;[244] it requires additional hardware which is not always available.

In our methods, we focused on the icosahedral atoms determined by Voronoi tessellation. Alternatively, we could use ML methods to discover relevant structure within the atomic configurations. In this case it may be necessary to define a specific task such as phase classification.[245] In addition, the per-atom methods we developed can be used in other applications.

6.3 Uncertainty for models and physical properties

We discussed the delta method, implemented it for a NN potential, and showed that it can determine input that extrapolates. For future work, the delta method can be included with on-the-fly MD. We can also extend the delta method for graph-NNs and experiment with approximations for large NNs.[136] Other research uses the loss Hessian for second order optimization methods for training.[246] The computational speed of delta method may also be increased by using Hessian-vector products.[157]

We contributed to deeper understanding of uncertainty obtained from different methods. We showed that Bayesian nonlinear regression and delta method uncertainties are consistent with each other, and their uncertainties are with respect to the specific model. We developed Gaussian process (GP) with first and second derivative information and showed GP variance includes model selection uncertainty. This is because GP posterior is a distribution of all reasonable models given the data. The developed GP framework can be used to train with derivative information. In future work, we can analyze Bayesian NNs' uncertainty and determine if it is model-general or model-specific. There are additional applications for uncertainty quantification with methods from Chapter 5 such as adsorption isotherms and spectroscopy. There are also many domain problems that can benefit from probabilistic graphical models, such as reaction rate determination, crack propagation in materials,

and process operations in chemical plants, as shown by our literature review in Section 5.1.2.

We look forward to continued advancement in these important problems around materials simulation.

References

[1] David Dubbeldam, Sofía Calero, Donald E. Ellis, and Randall Q. Snurr. RASPA: Molecular simulation software for adsorption and diffusion in flexible nanoporous materials. *Molecular Simulation*, 42(2):81–101, 2015. doi: 10.1080/08927022.2015.1010082. URL https://doi.org/10.1080/08927022.2015.1010082.

[2] Z. Guo, N. Saunders, A.P. Miodownik, and J.-Ph. Schillé. Modelling of materials properties and behaviour critical to casting simulation. *Materials Science and Engineering: A*, 413-414:465–469, 2005. doi: 10.1016/j.msea.2005.09.036. URL https://doi.org/10.1016/j.msea.2005.09.036.

[3] Liang-Feng Huang, Xue-Zeng Lu, Emrys Tennessen, and James M. Rondinelli. An efficient ab-initio quasiharmonic approach for the thermodynamics of solids. *Computational Materials Science*, 120:84–93, 2016. doi: 10.1016/j.commatsci.2016.04.012. URL https://doi.org/10.1016/j.commatsci.2016.04.012.

[4] Eric M. Lopato, Emily A. Eikey, Zoe C. Simon, Seoin Back, Kevin Tran, Jacqueline Lewis, Jakub F. Kowalewski, Sadegh Yazdi, John R. Kitchin, Zachary W. Ulissi, Jill E. Millstone, and Stefan Bernhard. Parallelized screening of characterized and DFT-modeled bimetallic colloidal cocatalysts for photocatalytic hydrogen evolution. *ACS Catalysis*, 10(7):4244–4252, 2020. doi: 10.1021/acscatal.9b05404. URL https://doi.org/10.1021/acscatal.9b05404.

[5] Kevin Tran and Zachary W. Ulissi. Active learning across intermetallics to guide discovery of electrocatalysts for CO_2 reduction and H_2 evolution. *Nature Catalysis*, 1(9):696–703, 2018. doi: 10.1038/s41929-018-0142-1. URL https://doi.org/10.1038/s41929-018-0142-1.

[6] Edward O. Pyzer-Knapp, Linjiang Chen, Graeme M. Day, and Andrew I. Cooper. Accelerating computational discovery of porous solids through improved navigation of energy-structure-function maps. *Science Advances*, 7(33), 2021. doi: 10.1126/sciadv.abi4763. URL https://doi.org/10.1126/sciadv.abi4763.

[7] Arunima K. Singh, Kiran Mathew, Houlong L. Zhuang, and Richard G. Hennig. Computational screening of 2D materials for photocatalysis. *The Journal of Physical Chemistry Letters*, 6(6):1087–1098, 2015. doi: 10.1021/jz502646d. URL https://doi.org/10.1021/jz502646d.

[8] Kyoungdoc Kim, Logan Ward, Jiangang He, Amar Krishna, Ankit Agrawal, and C. Wolverton. Machine-learning-accelerated high-throughput materials screening: Discovery of novel quaternary heusler compounds. *Physical Review Materials*, 2(12):123801, 2018. doi:

10.1103/physrevmaterials.2.123801. URL https://doi.org/10.1103/physrevmaterials.2.123801.

[9] Karlheinz Schwarz and Peter Blaha. *DFT Calculations for Real Solids*, pages 227–259. Handbook of Solid State Chemistry. Wiley-VCH Verlag GmbH & Co. KGaA, 2017. doi: 10.1002/9783527691036.hsscvol5022. URL https://doi.org/10.1002/9783527691036.hsscvol5022.

[10] Errol G. Lewars. *The Concept of the Potential Energy Surface*, pages 9–49. Springer International Publishing, Cham, 2016. ISBN 978-3-319-30916-3. doi: 10.1007/978-3-319-30916-3_2. URL https://doi.org/10.1007/978-3-319-30916-3_2.

[11] David Sholl and Janice A Steckel. *Density functional theory: A practical introduction*. John Wiley & Sons, 2011. doi: 10.1002/9780470447710.

[12] Murray S. Daw and M. I. Baskes. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Physical Review B*, 29(12):6443–6453, 1984. doi: 10.1103/physrevb.29.6443. URL https://doi.org/10.1103/physrevb.29.6443.

[13] Adri C. T. van Duin, Siddharth Dasgupta, Francois Lorant, and William A. Goddard. ReaxFF: A reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, 105(41):9396–9409, 2001. doi: 10.1021/jp004368u. URL https://doi.org/10.1021/jp004368u.

[14] S. Starikov, I. Gordeev, Y. Lysogorskiy, L. Kolotova, and S. Makarov. Optimized interatomic potential for study of structure and phase transitions in Si-Au and Si-Al systems. *Computational Materials Science*, 184:109891, 2020. doi: 10.1016/j.commatsci.2020.109891. URL https://doi.org/10.1016/j.commatsci.2020.109891.

[15] F Ercolessi and J. B Adams. Interatomic potentials from first-principles calculations: The force-matching method. *Europhysics Letters (EPL)*, 26(8):583–588, 1994. doi: 10.1209/0295-5075/26/8/005. URL https://doi.org/10.1209/0295-5075/26/8/005.

[16] C A Howells and Y Mishin. Angular-dependent interatomic potential for the binary Ni-Cr system. *Modelling and Simulation in Materials Science and Engineering*, 26(8):085008, 2018. doi: 10.1088/1361-651x/aae400. URL https://doi.org/10.1088/1361-651x/aae400.

[17] John R. Kitchin. Machine learning in catalysis. *Nature Catalysis*, 1(4):230–232, 2018. doi: 10.1038/s41929-018-0056-y. URL https://doi.org/10.1038/s41929-018-0056-y.

[18] Mark E. Tuckerman, Yi Liu, Giovanni Ciccotti, and Glenn J. Martyna. Non-Hamiltonian molecular dynamics: Generalizing Hamiltonian phase

space principles to non-Hamiltonian systems. *The Journal of Chemical Physics*, 115(4):1678–1702, 2001. doi: 10.1063/1.1378321. URL https://doi.org/10.1063/1.1378321.

[19] M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publ. Clarendon Press, 1989. ISBN 9780198556459. URL https://books.google.com/books?id=O32VXB9e5P4C.

[20] William C. Swope, Hans C. Andersen, Peter H. Berens, and Kent R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics*, 76(1):637–649, 1982. doi: 10.1063/1.442716. URL https://doi.org/10.1063/1.442716.

[21] Glenn J. Martyna, Michael L. Klein, and Mark Tuckerman. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics*, 97(4):2635–2643, 1992. doi: 10.1063/1.463940. URL https://doi.org/10.1063/1.463940.

[22] Mark E Tuckerman, José Alejandre, Roberto López-Rendón, Andrea L Jochim, and Glenn J Martyna. A Liouville-operator derived measure-preserving integrator for molecular dynamics simulations in the isothermal-isobaric ensemble. *Journal of Physics A: Mathematical and General*, 39(19):5629–5651, 2006. doi: 10.1088/0305-4470/39/19/s18. URL https://doi.org/10.1088/0305-4470/39/19/s18.

[23] Glenn J. Martyna, Douglas J. Tobias, and Michael L. Klein. Constant pressure molecular dynamics algorithms. *The Journal of Chemical Physics*, 101(5):4177–4189, 1994. doi: 10.1063/1.467468. URL https://doi.org/10.1063/1.467468.

[24] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1), Jan 2010. ISSN 0965-0393. doi: 10.1088/0965-0393/18/1/015012.

[25] Tom Mitchell. Machine learning. 1997.

[26] Samad Hajinazar, Junping Shao, and Aleksey N. Kolmogorov. Stratified construction of neural network based interatomic models for multicomponent materials. *Physical Review B*, 95(1):014114, 2017. doi: 10.1103/physrevb.95.014114. URL https://doi.org/10.1103/physrevb.95.014114.

[27] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. doi:

10.1016/0893-6080(91)90009-t. URL https://doi.org/10.1016/0893-6080(91)90009-t.

[28] Christopher Woodward, Mark Asta, Dallas R. Trinkle, James Lill, and Stefano Angioletti-Uberti. Ab initio simulations of molten Ni alloys. *Journal of Applied Physics*, 107(11):113522, 2010. doi: 10.1063/1.3437644. URL https://doi.org/10.1063/1.3437644.

[29] C. Woodward, M. Asta, D. R. Trinkle, J. Lill, and S. Angioletti-Uberti. Ab-initio molecular dynamics simulations of molten Ni-based superalloys. In *2008 DoD HPCMP Users Group Conference*, pages 169–174, July 2008. doi: 10.1109/DoD.HPCMP.UGC.2008.15.

[30] C. Woodward, D. Trinkle, and M. Asta. Ab initio molecular dynamics simulations of molten Ni-based superalloys. In *HPCMP Users Group Conference (HPCMP-UGC)*, volume 00, pages 147–152, 06 2007. doi: 10.1109/HPCMP-UGC.2007.1. URL doi.ieeecomputersociety.org/10.1109/HPCMP-UGC.2007.1.

[31] Christopher Woodward, James Lill, Mark Asta, and Dallas R Trinkle. Molecular-dynamics simulations of molten Ni-based superalloys. *Superalloys 2012*, pages 537–545, 2012. doi: 10.7449/2012/Superalloys_2012_537_545.

[32] D. Trinkle, M. Asta, and C. Woodward. Ab-initio molecular dynamics simulations of molten Ni-based superalloys. In *HPCMP Users Group Conference (HPCMP-UGC)*, volume 00, pages 177–181, 06 2006. doi: 10.1109/HPCMP-UGC.2006.1. URL doi.ieeecomputersociety.org/10.1109/HPCMP-UGC.2006.1.

[33] Jörg Behler. First principles neural network potentials for reactive simulations of large molecular and condensed systems. *Angewandte Chemie International Edition*, 56(42):12828–12840, 2017. doi: 10.1002/anie.201703114. URL https://doi.org/10.1002/anie.201703114.

[34] J. S. Smith, O. Isayev, and A. E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science*, 8(4):3192–3203, 2017. doi: 10.1039/c6sc05720a. URL https://doi.org/10.1039/c6sc05720a.

[35] Jacob R. Boes and John R. Kitchin. Neural network predictions of oxygen interactions on a dynamic Pd surface. *Molecular Simulation*, 43 (5-6):346–354, 2017. doi: 10.1080/08927022.2016.1274984. URL https://doi.org/10.1080/08927022.2016.1274984.

[36] Khosrow Shakouri, Jörg Behler, Jörg Meyer, and Geert-Jan Kroes. Accurate neural network description of surface phonons in reactive gas-surface dynamics: N_2 + Ru(0001). *The Journal of Physical Chemistry*

Letters, 8(10):2131–2136, 2017. doi: 10.1021/acs.jpclett.7b00784. URL https://doi.org/10.1021/acs.jpclett.7b00784.

[37] Ryo Kobayashi, Daniele Giofré, Till Junge, Michele Ceriotti, and William A. Curtin. Neural network potential for Al-Mg-Si alloys. *Physical Review Materials*, 1(5):053604, 2017. doi: 10.1103/physrevmaterials.1.053604. URL https://doi.org/10.1103/physrevmaterials.1.053604.

[38] Volker L. Deringer, Noam Bernstein, Albert P. Bartók, Matthew J. Cliffe, Rachel N. Kerber, Lauren E. Marbella, Clare P. Grey, Stephen R. Elliott, and Gábor Csányi. Realistic atomistic structure of amorphous silicon from machine-learning-driven molecular dynamics. *The Journal of Physical Chemistry Letters*, 9(11):2879–2885, 2018. doi: 10.1021/acs.jpclett.8b00902. URL https://doi.org/10.1021/acs.jpclett.8b00902.

[39] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98(14):146401, 2007. doi: 10.1103/physrevlett.98.146401. URL https://doi.org/10.1103/physrevlett.98.146401.

[40] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104(13):136403, 2010. doi: 10.1103/physrevlett.104.136403. URL https://doi.org/10.1103/physrevlett.104.136403.

[41] Atsuto Seko, Akira Takahashi, and Isao Tanaka. Sparse representation for a potential energy surface. *Physical Review B*, 90(2):024101, 2014. doi: 10.1103/physrevb.90.024101. URL https://doi.org/10.1103/physrevb.90.024101.

[42] Venkatesh Botu and Rampi Ramprasad. Adaptive machine learning framework to accelerate ab initio molecular dynamics. *International Journal of Quantum Chemistry*, 115(16):1074–1083, 2014. doi: 10.1002/qua.24836. URL https://doi.org/10.1002/qua.24836.

[43] Mardochee Reveil and Paulette Clancy. Classification of spatially resolved molecular fingerprints for machine learning applications and development of a codebase for their implementation. *Molecular Systems Design & Engineering*, 3:431–441, 2018. doi: 10.1039/c8me00003d. URL https://doi.org/10.1039/c8me00003d.

[44] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. *International Journal of Quantum Chemistry*, 115(16):1032–1050, 2015. doi: 10.1002/qua.24890. URL https://doi.org/10.1002/qua.24890.

[45] Tran Doan Huan, Rohit Batra, James Chapman, Sridevi Krishnan, Lihua Chen, and Rampi Ramprasad. A universal strategy for the creation of machine learning-based atomistic force fields. *npj Computational Materials*, 3(1):37, 2017. doi: 10.1038/s41524-017-0042-y. URL https://doi.org/10.1038/s41524-017-0042-y.

[46] Volker L. Deringer and Gábor Csányi. Machine learning based interatomic potential for amorphous carbon. *Physical Review B*, 95 (9):094203, 2017. doi: 10.1103/physrevb.95.094203. URL https://doi.org/10.1103/physrevb.95.094203.

[47] Albert P. Bartók, James Kermode, Noam Bernstein, and Gábor Csányi. Machine learning a general-purpose interatomic potential for silicon. *Physical Review X*, 8(4):041048, 2018. doi: 10.1103/physrevx.8.041048. URL https://doi.org/10.1103/physrevx.8.041048.

[48] Volker L. Deringer, Chris J. Pickard, and Gábor Csányi. Data-driven learning of total and local energies in elemental boron. *Physical Review Letters*, 120(15):156001, 2018. doi: 10.1103/physrevlett.120.156001. URL https://doi.org/10.1103/physrevlett.120.156001.

[49] Justin S. Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E. Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24):241733, 2018. doi: 10.1063/1.5023802. URL https://doi.org/10.1063/1.5023802.

[50] Li Zhu, Maximilian Amsler, Tobias Fuhrer, Bastian Schaefer, Somayeh Faraji, Samare Rostami, S. Alireza Ghasemi, Ali Sadeghi, Migle Grauzinyte, Chris Wolverton, and Stefan Goedecker. A fingerprint based metric for measuring similarities of crystalline structures. *The Journal of Chemical Physics*, 144(3):034203, 2016. doi: 10.1063/1.4940026. URL https://doi.org/10.1063/1.4940026.

[51] Sandip De, Albert P. Bartók, Gábor Csányi, and Michele Ceriotti. Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics*, 18(20):13754–13769, 2016. doi: 10.1039/c6cp00415f. URL https://doi.org/10.1039/c6cp00415f.

[52] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013. doi: 10.1103/physrevb.87.184115. URL https://doi.org/10.1103/physrevb.87.184115.

[53] Piero Gasparotto, Robert Horst Meißner, and Michele Ceriotti. Recognizing local and global structural motifs at the atomic scale. *Journal of Chemical Theory and Computation*, 14(2):486–498, 2018. doi:

10.1021/acs.jctc.7b00993. URL https://doi.org/10.1021/acs.jctc.7b00993.

[54] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[55] Ekin D. Cubuk, Brad D. Malone, Berk Onat, Amos Waterland, and Efthimios Kaxiras. Representations in neural network based empirical potentials. *The Journal of Chemical Physics*, 147(2):024104, 2017. doi: 10.1063/1.4990503. URL https://doi.org/10.1063/1.4990503.

[56] Michele Ceriotti, Gareth A. Tribello, and Michele Parrinello. Simplifying the representation of complex free-energy landscapes using sketch-map. *Proceedings of the National Academy of Sciences*, 108(32):13023–13028, 2011. doi: 10.1073/pnas.1108486108. URL https://doi.org/10.1073/pnas.1108486108.

[57] Michele Ceriotti, Gareth A. Tribello, and Michele Parrinello. Demonstrating the transferability and the descriptive power of sketch-map. *Journal of Chemical Theory and Computation*, 9(3):1521–1532, 2013. doi: 10.1021/ct3010563. URL https://doi.org/10.1021/ct3010563.

[58] Sönke Lorenz, Matthias Scheffler, and Axel Gross. Descriptions of surface chemical reactions using a neural network representation of the potential-energy surface. *Physical Review B*, 73(11):115431, 2006. doi: 10.1103/physrevb.73.115431. URL https://doi.org/10.1103/physrevb.73.115431.

[59] Félix Musil, Sandip De, Jack Yang, Joshua E. Campbell, Graeme M. Day, and Michele Ceriotti. Machine learning for the structure-energy-property landscapes of molecular crystals. *Chemical Science*, 9(5):1289–1300, 2018. doi: 10.1039/c7sc04665k. URL https://doi.org/10.1039/c7sc04665k.

[60] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996. doi: 10.1103/physrevb.54.11169. URL https://doi.org/10.1103/physrevb.54.11169.

[61] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *CoRR*, 2016. URL http://arxiv.org/abs/1603.06560v4.

[62] A. Pasturel and N. Jakse. Short-range structural signature of transport properties of Al-Ni melts. *Journal of Non-Crystalline Solids*, 425:176–182, 2015. doi: 10.1016/j.jnoncrysol.2015.06.014. URL https://doi.org/10.1016/j.jnoncrysol.2015.06.014.

[63] James R. Morris, U. Dahlborg, and M. Calvo-Dahlborg. Recent developments and outstanding challenges in theory and modeling of liquid metals. *Journal of Non-Crystalline Solids*, 353(32-40):3444–3453, 2007. doi: 10.1016/j.jnoncrysol.2007.05.159. URL https://doi.org/10.1016/j.jnoncrysol.2007.05.159.

[64] Kenneth R. Harris. The fractional Stokes-Einstein equation: Application to Lennard-Jones, molecular, and ionic liquids. *The Journal of Chemical Physics*, 131(5):054503, 2009. doi: 10.1063/1.3183951. URL https://doi.org/10.1063/1.3183951.

[65] Vikas Dubey, Shakkira Erimban, Sandipa Indra, and Snehasis Daschakraborty. Understanding the origin of the breakdown of the Stokes-Einstein relation in supercooled water at different temperature-pressure conditions. *The Journal of Physical Chemistry B*, 123(47): 10089–10099, 2019. doi: 10.1021/acs.jpcb.9b08309. URL https://doi.org/10.1021/acs.jpcb.9b08309.

[66] Noel Jakse and Alain Pasturel. Liquid aluminum: Atomic diffusion and viscosity from ab initio molecular dynamics. *Scientific Reports*, 3(1): 3135, 2013. doi: 10.1038/srep03135. URL https://doi.org/10.1038/srep03135.

[67] M. Trybula, N. Jakse, W. Gasior, and A. Pasturel. Structural and physicochemical properties of liquid Al-Zn alloys: A combined study based on molecular dynamics simulations and the quasi-lattice theory. *The Journal of Chemical Physics*, 141(22):224504, 2014. doi: 10.1063/1.4903209. URL https://doi.org/10.1063/1.4903209.

[68] Markus M. Hoffmann, Matthew D. Too, Michael Vogel, Torsten Gutmann, and Gerd Buntkowsky. Breakdown of the Stokes-Einstein equation for solutions of water in oil reverse micelles. *The Journal of Physical Chemistry B*, 124(41):9115–9125, 2020. doi: 10.1021/acs.jpcb.0c06124. URL https://doi.org/10.1021/acs.jpcb.0c06124.

[69] A Pasturel and N Jakse. On the role of entropy in determining transport properties in metallic melts. *Journal of Physics: Condensed Matter*, 27 (32):325104, 2015. doi: 10.1088/0953-8984/27/32/325104. URL https://doi.org/10.1088/0953-8984/27/32/325104.

[70] Lorenzo Costigliola, David M. Heyes, Thomas B. Schrøder, and Jeppe C. Dyre. Revisiting the Stokes-Einstein relation without a hydrodynamic

diameter. *The Journal of Chemical Physics*, 150(2):021101, 2019. doi: 10.1063/1.5080662. URL https://doi.org/10.1063/1.5080662.

[71] C.H. Li, Y.W. Luan, X.J. Han, and J.G. Li. Structural aspects of the Stokes-Einstein relation breakdown in high temperature melts. *Journal of Non-Crystalline Solids*, 458:107–117, 2017. doi: 10.1016/j.jnoncrysol.2016.12.025. URL https://doi.org/10.1016/j.jnoncrysol.2016.12.025.

[72] Y.H. Zhou, X.J. Han, and J.G. Li. Transport properties and abnormal breakdown of the Stokes-Einstein relation in computer simulated $Al_{72}Ni_{16}Co_{12}$ metallic melt. *Journal of Non-Crystalline Solids*, 517:83–95, 2019. doi: 10.1016/j.jnoncrysol.2019.04.035. URL https://doi.org/10.1016/j.jnoncrysol.2019.04.035.

[73] Y. C. Hu, F. X. Li, M. Z. Li, H. Y. Bai, and W. H. Wang. Structural signatures evidenced in dynamic crossover phenomena in metallic glass-forming liquids. *Journal of Applied Physics*, 119(20):205108, 2016. doi: 10.1063/1.4952986. URL https://doi.org/10.1063/1.4952986.

[74] N. Jakse and A. Pasturel. Transport properties and Stokes-Einstein relation in Al-rich liquid alloys. *The Journal of Chemical Physics*, 144 (24):244502, 2016. doi: 10.1063/1.4954322. URL https://doi.org/10.1063/1.4954322.

[75] Christina Cruickshank Miller. The Stokes-Einstein law for diffusion in solution. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(740): 724–749, 1924. doi: 10.1098/rspa.1924.0100. URL https://doi.org/10.1098/rspa.1924.0100.

[76] Chen-Hui Li, Xiu-Jun Han, Ying-Wei Luan, and Jian-Guo Li. Abnormal breakdown of Stokes-Einstein relation in liquid aluminium. *Chinese Physics B*, 26(1):016102, 2017. doi: 10.1088/1674-1056/26/1/016102. URL https://doi.org/10.1088/1674-1056/26/1/016102.

[77] Frans Spaepen. Five-fold symmetry in liquids. *Nature*, 408(6814):781–782, 2000. doi: 10.1038/35048652. URL https://doi.org/10.1038/35048652.

[78] Y. C. Hu, F. X. Li, M. Z. Li, H. Y. Bai, and W. H. Wang. Five-fold symmetry as indicator of dynamic arrest in metallic glass-forming liquids. *Nature Communications*, 6(1):8310, 2015. doi: 10.1038/ncomms9310. URL https://doi.org/10.1038/ncomms9310.

[79] Songyou Wang, C. Z. Wang, Feng-Chuan Chuang, James R. Morris, and K. M. Ho. Ab initio molecular dynamics simulation of liquid $Al_{88}Si_{12}$

alloys. *The Journal of Chemical Physics*, 122(3):034508, 2005. doi: 10.1063/1.1833355. URL https://doi.org/10.1063/1.1833355.

[80] K. H. Khoo, T.-L. Chan, M. Kim, and James R. Chelikowsky. Ab initio molecular dynamics simulations of molten $Al_{1-x}Si_x$ alloys. *Physical Review B*, 84(21):214203, 2011. doi: 10.1103/physrevb.84.214203. URL https://doi.org/10.1103/physrevb.84.214203.

[81] Jingyu Qin, Shaopeng Pan, Yuanhua Qi, and Tingkun Gu. The structure and thermodynamic properties of liquid Al-Si alloys by ab initio molecular dynamics simulation. *Journal of Non-Crystalline Solids*, 433:31–37, 2016. doi: 10.1016/j.jnoncrysol.2015.11.032. URL https://doi.org/10.1016/j.jnoncrysol.2015.11.032.

[82] Jingyu Qin, Xinxin Li, Jin Wang, and Shaopeng Pan. The self-diffusion coefficients of liquid binary M-Si (M=Al, Fe, Mg and Au) alloy systems by first principles molecular dynamics simulation. *AIP Advances*, 9(3): 035328, 2019. doi: 10.1063/1.5067295. URL https://doi.org/10.1063/1.5067295.

[83] Venkateswara Rao Manga and D R Poirier. Ab initio molecular dynamics simulation of self-diffusion in Al-Si binary melts. *Modelling and Simulation in Materials Science and Engineering*, 26(6):065006, 2018. doi: 10.1088/1361-651x/aacdbc. URL https://doi.org/10.1088/1361-651x/aacdbc.

[84] M Ji and X G Gong. Ab initio molecular dynamics simulation on temperature-dependent properties of Al-Si liquid alloy. *Journal of Physics: Condensed Matter*, 16(15):2507–2514, 2004. doi: 10.1088/0953-8984/16/15/004. URL https://doi.org/10.1088/0953-8984/16/15/004.

[85] Xiusong Huang, Xixi Dong, Lehua Liu, and Peijie Li. Liquid structure of Al-Si alloy: A molecular dynamics simulation. *Journal of Non-Crystalline Solids*, 503-504:182–185, 2019. doi: 10.1016/j.jnoncrysol.2018.09.047. URL https://doi.org/10.1016/j.jnoncrysol.2018.09.047.

[86] P. Saidi and J.J. Hoyt. Atomistic simulation of the step mobility at the Al-Si(1 1 1) crystal-melt interface using molecular dynamics. *Computational Materials Science*, 111:137–147, 2016. doi: 10.1016/j.commatsci.2015.09.040. URL https://doi.org/10.1016/j.commatsci.2015.09.040.

[87] P. Ganesh and M. Widom. Signature of nearly icosahedral structures in liquid and supercooled liquid copper. *Physical Review B*, 74(13):134205, 2006. doi: 10.1103/physrevb.74.134205. URL https://doi.org/10.1103/physrevb.74.134205.

[88] P. Ganesh and M. Widom. Ab initio simulations of geometrical frustration in supercooled liquid Fe and Fe-based metallic glass. *Physical Review B*, 77(1):014205, 2008. doi: 10.1103/physrevb.77.014205. URL https://doi.org/10.1103/physrevb.77.014205.

[89] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995. doi: 10.1006/jcph.1995.1039. URL https://doi.org/10.1006/jcph.1995.1039.

[90] Nicholas A. Nystrom, Michael J. Levine, Ralph Z. Roskies, and J. Ray Scott. Bridges. In *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15*, 2015. doi: 10.1145/2792745.2792775. URL http://dx.doi.org/10.1145/2792745.2792775.

[91] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. XSEDE: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, 2014. doi: 10.1109/mcse.2014.80. URL http://dx.doi.org/10.1109/MCSE.2014.80.

[92] Ni Zhan and John R. Kitchin. Data and code: Origin of the Stokes-Einstein deviation in liquid Al-Si https://doi.org/10.5281/zenodo.5554968, 2021.

[93] Edward J. Maginn, Richard A. Messerly, Daniel J. Carlson, Daniel R. Roe, and J. Richard Elliott. Best practices for computing transport properties 1. Self-diffusivity and viscosity from equilibrium molecular dynamics [article v1.0]. *Living Journal of Computational Molecular Science*, 1(1), 2019. doi: 10.33011/livecoms.1.1.6324. URL https://doi.org/10.33011/livecoms.1.1.6324.

[94] In-Chul Yeh and Gerhard Hummer. System-size dependence of diffusion coefficients and viscosities from molecular dynamics simulations with periodic boundary conditions. *The Journal of Physical Chemistry B*, 108(40):15873–15879, 2004. doi: 10.1021/jp0477147. URL https://doi.org/10.1021/jp0477147.

[95] E M Kirova and G E Norman. Viscosity calculations at molecular dynamics simulations. *Journal of Physics: Conference Series*, 653: 012106, 2015. doi: 10.1088/1742-6596/653/1/012106. URL https://doi.org/10.1088/1742-6596/653/1/012106.

[96] Guang-Jun Guo, Yi-Gang Zhang, Keith Refson, and Ya-Juan Zhao. Viscosity and stress autocorrelation function in supercooled water: A

molecular dynamics study. *Molecular Physics*, 100(16):2617–2627, 2002. doi: 10.1080/00268970210133477. URL https://doi.org/10.1080/00268970210133477.

[97] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[98] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[99] Lili Ju, Todd Ringler, and Max Gunzburger. *Voronoi Tessellations and Their Application to Climate and Global Modeling*, pages 313–342. Numerical Techniques for Global Atmospheric Models. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-11640-7_10. URL https://doi.org/10.1007/978-3-642-11640-7_10.

[100] F. Li, X.J. Liu, and Z.P. Lu. Atomic structural evolution during glass formation of a Cu-Zr binary metallic glass. *Computational Materials Science*, 85:147–153, 2014. doi: 10.1016/j.commatsci.2013.12.058. URL https://doi.org/10.1016/j.commatsci.2013.12.058.

[101] S. Trady, A. Hasnaoui, and M. Mazroui. Atomic packing and medium-range order in Ni_3Al metallic glass. *Journal of Non-Crystalline Solids*, 468:27–33, 2017. doi: 10.1016/j.jnoncrysol.2017.04.026. URL https://doi.org/10.1016/j.jnoncrysol.2017.04.026.

[102] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of*

Machine Learning Research, 12:2825–2830, 2011. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[103] Diego Coglitore, Stuart P. Edwardson, Peter Macko, Eann A. Patterson, and Maurice Whelan. Transition from fractional to classical Stokes-Einstein behaviour in simple fluids. *Royal Society Open Science*, 4(12): 170507, 2017. doi: 10.1098/rsos.170507. URL https://doi.org/10.1098/rsos.170507.

[104] Shaopeng Pan, Z. W. Wu, W. H. Wang, M. Z. Li, and Limei Xu. Structural origin of fractional Stokes-Einstein relation in glass-forming liquids. *Scientific Reports*, 7(1):39938, 2017. doi: 10.1038/srep39938. URL https://doi.org/10.1038/srep39938.

[105] Aidan P. Thompson, Steven J. Plimpton, and William Mattson. General formulation of pressure and stress tensor for arbitrary many-body interaction potentials under periodic boundary conditions. *The Journal of Chemical Physics*, 131(15):154107, 2009. doi: 10.1063/1.3245303. URL https://doi.org/10.1063/1.3245303.

[106] M. J. Assael, E. K. Mihailidou, J. Brillo, S. V. Stankus, J. T. Wu, and W. A. Wakeham. Reference correlation for the density and viscosity of eutectic liquid alloys Al+Si, Pb+Bi, and Pb+Sn. *Journal of Physical and Chemical Reference Data*, 41(3):033103, 2012. doi: 10.1063/1.4750035. URL https://doi.org/10.1063/1.4750035.

[107] X.J. Liu, Y. Xu, Z.P. Lu, X. Hui, G.L. Chen, G.P. Zheng, and C.T. Liu. Atomic packing symmetry in the metallic liquid and glass states. *Acta Materialia*, 59(16):6480–6488, 2011. doi: 10.1016/j.actamat.2011.07.012. URL https://doi.org/10.1016/j.actamat.2011.07.012.

[108] S. P. Pan, J. Y. Qin, W. M. Wang, and T. K. Gu. Origin of splitting of the second peak in the pair-distribution function for metallic glasses. *Physical Review B*, 84(9):092201, 2011. doi: 10.1103/physrevb.84.092201. URL https://doi.org/10.1103/physrevb.84.092201.

[109] Jun Ding, Evan Ma, Mark Asta, and Robert O. Ritchie. Second-nearest-neighbor correlations from connection of atomic packing motifs in metallic glasses and liquids. *Scientific Reports*, 5(1):17429, 2015. doi: 10.1038/srep17429. URL https://doi.org/10.1038/srep17429.

[110] M Hoffmann, A Marmodoro, A Ernst, W Hergert, J Dahl, J Lång, P Laukkanen, M P J Punkkinen, and K Kokko. Quantitative description of short-range order and its influence on the electronic structure in Ag-Pd alloys. *Journal of Physics: Condensed Matter*, 28(30):305501, 2016. doi: 10.1088/0953-8984/28/30/305501. URL https://doi.org/10.1088/0953-8984/28/30/305501.

[111] A. Fernandez-Caballero, J. S. Wrobel, P. M. Mummery, and D. Nguyen-Manh. Short-range order in high entropy alloys: Theoretical formulation and application to Mo-Nb-Ta-V-W system. *CoRR*, 2017. URL http://arxiv.org/abs/1705.01844v1.

[112] G. Q. Yue, Y. Zhang, Y. Sun, B. Shen, F. Dong, Z. Y. Wang, R. J. Zhang, Y. X. Zheng, M. J. Kramer, S. Y. Wang, C. Z. Wang, K. M. Ho, and L. Y. Chen. Local structure order in $Pd_{78}Cu_6Si_{16}$ liquid. *Scientific Reports*, 5(1):8277, 2015. doi: 10.1038/srep08277. URL https://doi.org/10.1038/srep08277.

[113] Abraham. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36 (8):1627–1639, 1964. doi: 10.1021/ac60214a047. URL https://doi.org/10.1021/ac60214a047.

[114] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *CoRR*, 2019. URL http://arxiv.org/abs/1906.02530v2.

[115] Benjamin Kompa, Jasper Snoek, and Andrew L. Beam. Second opinion needed: Communicating uncertainty in medical machine learning. *npj Digital Medicine*, 4(1):4, 2021. doi: 10.1038/s41746-020-00367-3. URL https://doi.org/10.1038/s41746-020-00367-3.

[116] Larry Wasserman. *All of Statistics*. Springer Texts in Statistics. Springer New York, 2004. doi: 10.1007/978-0-387-21736-9. URL https://doi.org/10.1007/978-0-387-21736-9.

[117] Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. *CoRR*, 2019. URL http://arxiv.org/abs/1902.02476v2.

[118] Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 1169–1179, Tel Aviv, Israel, 22–25 Jul 2020. PMLR. URL http://proceedings.mlr.press/v115/izmailov20a.html.

[119] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *CoRR*, 2020. URL http://arxiv.org/abs/2006.10108v2.

[120] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *CoRR*, 2016. URL http://arxiv.org/abs/1604.04173v2.

[121] Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. *CoRR*, 2019. URL http://arxiv.org/abs/1905.03222v1.

[122] Tomohiro Endo, Tomoaki Watanabe, and Akio Yamamoto. Confidence interval estimation by bootstrap method for uncertainty quantification using random sampling method. *Journal of Nuclear Science and Technology*, 52(7-8):993–999, 2015. doi: 10.1080/00223131.2015.1034216. URL https://doi.org/10.1080/00223131.2015.1034216.

[123] Glenn Palmer, Siqi Du, Alexander Politowicz, Joshua Paul Emory, Xiyu Yang, Anupraas Gautam, Grishma Gupta, Zhelong Li, Ryan Jacobs, and Dane Morgan. Calibrated bootstrap for uncertainty quantification in regression models. *CoRR*, 2021. URL http://arxiv.org/abs/2105.13303v1.

[124] Hongfei Du, Emre Barut, and Fang Jin. Uncertainty quantification in cnn through the bootstrap of convex neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):12078–12085, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17434.

[125] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021. doi: 10.1016/j.inffus.2021.05.008. URL https://doi.org/10.1016/j.inffus.2021.05.008.

[126] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *CoRR*, 2020. URL http://arxiv.org/abs/2006.13570v3.

[127] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *CoRR*, 2016. URL http://arxiv.org/abs/1612.01474v3.

[128] James O. Berger and Leonard A. Smith. On the statistical formalism of uncertainty quantification. *Annual Review of Statistics and Its Application*, 6(1):433–460, 2019. doi: 10.1146/annurev-statistics-030718-105232. URL https://doi.org/10.1146/annurev-statistics-030718-105232.

[129] Robert Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8(1):152–163, 1996. doi: 10.1162/neco.1996.8.1.152. URL https://doi.org/10.1162/neco.1996.8.1.152.

[130] Richard Dybowski and Stephen J Roberts. Confidence intervals and prediction intervals for feed-forward neural networks. *Clinical Applications of Artificial Neural Networks*, pages 298–326, 2001.

[131] CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL www.GaussianProcess.org/gpml.

[132] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *CoRR*, 2015. URL http://arxiv.org/abs/1506.02142v6.

[133] David John Cameron Mackay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[134] Radford M. Neal. *Priors for Infinite Networks*, pages 29–53. Bayesian Learning for Neural Networks. Springer New York, 1996. doi: 10.1007/978-1-4612-0745-0_2. URL https://doi.org/10.1007/978-1-4612-0745-0_2.

[135] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *CoRR*, 2015. URL http://arxiv.org/abs/1505.05424v2.

[136] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[137] Huijie Tian, Christopher Rzepa, Ronak Upadhyay, and Srinivas Rangarajan. Estimating vibrational and thermodynamic properties of adsorbates with uncertainty using data driven surrogates. *AIChE Journal*, 65(12), 2019. doi: 10.1002/aic.16838. URL https://doi.org/10.1002/aic.16838.

[138] Janet R. Donaldson and Robert B. Schnabel. Computational experience with confidence regions and confidence intervals for nonlinear least squares. *Technometrics*, 29(1):67–82, 1987. doi: 10.1080/00401706.1987.10488184. URL https://doi.org/10.1080/00401706.1987.10488184.

[139] Richard D. de Veaux, Jennifer Schumi, Jason Schweinsberg, and Lyle H. Ungar. Prediction intervals for neural networks via nonlinear regression.

Technometrics, 40(4):273, 1998. doi: 10.2307/1270528. URL https://doi.org/10.2307/1270528.

[140] G. Papadopoulos, P.J. Edwards, and A.F. Murray. Confidence estimation methods for neural networks: A practical comparison. *IEEE Transactions on Neural Networks*, 12(6):1278–1287, 2001. doi: 10.1109/72.963764. URL https://doi.org/10.1109/72.963764.

[141] Burr Settles. Active learning literature survey. 2009.

[142] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks*, 22(9):1341–1356, 2011. doi: 10.1109/tnn.2011.2162110. URL https://doi.org/10.1109/tnn.2011.2162110.

[143] J Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *Journal of Physics: Condensed Matter*, 26 (18):183001, 2014. doi: 10.1088/0953-8984/26/18/183001. URL https://doi.org/10.1088/0953-8984/26/18/183001.

[144] Andrew A. Peterson, Rune Christensen, and Alireza Khorshidi. Addressing uncertainty in atomistic machine learning. *Physical Chemistry Chemical Physics*, 19(18):10978–10985, 2017. doi: 10.1039/c7cp00375g. URL https://doi.org/10.1039/c7cp00375g.

[145] Justin S Smith, Benjamin T. Nebgen, Roman Zubatyuk, Nicholas Lubbers, Christian Devereux, Kipton Barros, Sergei Tretiak, Olexandr Isayev, and Adrian Roitberg. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. 6 2019. doi: 10.26434/chemrxiv.6744440.v2. URL https://chemrxiv.org/articles/Outsmarting_Quantum_Chemistry_Through_Transfer_Learning/6744440.

[146] Zhenwei Li, James R. Kermode, and Alessandro De Vita. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Physical Review Letters*, 114(9):096405, 2015. doi: 10.1103/physrevlett.114.096405. URL https://doi.org/10.1103/physrevlett.114.096405.

[147] Nongnuch Artrith and Jörg Behler. High-dimensional neural network potentials for metal surfaces: A prototype study for copper. *Physical Review B*, 85(4):045439, 2012. doi: 10.1103/physrevb.85.045439. URL https://doi.org/10.1103/physrevb.85.045439.

[148] Jonathan Vandermause, Steven B. Torrisi, Simon Batzner, Yu Xie, Lixin Sun, Alexie M. Kolpak, and Boris Kozinsky. On-the-fly active learning of interpretable Bayesian force fields for

atomistic rare events. *npj Computational Materials*, 6(1):20, 2020. doi: 10.1038/s41524-020-0283-z. URL https://doi.org/10.1038/s41524-020-0283-z.

[149] Yu Xie, Jonathan Vandermause, Lixin Sun, Andrea Cepellotti, and Boris Kozinsky. Bayesian force fields from active learning for simulation of inter-dimensional transformation of stanene. *npj Computational Materials*, 7(1):40, 2021. doi: 10.1038/s41524-021-00510-y. URL https://doi.org/10.1038/s41524-021-00510-y.

[150] Rune Christensen. *Error Mitigation in Computational Design of Sustainable Energy Materials*. PhD thesis, Department of Energy Conversion and Storage, Technical University of Denmark, 2016.

[151] Mingjian Wen and Ellad B. Tadmor. Uncertainty quantification in molecular simulations with dropout neural network potentials. *npj Computational Materials*, 6(1):124, 2020. doi: 10.1038/s41524-020-00390-8. URL https://doi.org/10.1038/s41524-020-00390-8.

[152] Jon Paul Janet, Chenru Duan, Tzuhsiung Yang, Aditya Nandy, and Heather Kulik. A quantitative uncertainty metric controls error in neural network-driven chemical discovery. *Chemical Science*, 2019. doi: 10.1039/c9sc02298h. URL https://doi.org/10.1039/c9sc02298h.

[153] Kevin Tran, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W Ulissi. Methods for comparing uncertainty quantifications for material property predictions. *Machine Learning: Science and Technology*, 2020. doi: 10.1088/2632-2153/ab7e1a. URL https://doi.org/10.1088/2632-2153/ab7e1a.

[154] Félix Musil, Michael J. Willatt, Mikhail A. Langovoy, and Michele Ceriotti. Fast and accurate uncertainty estimation in chemical machine learning. *Journal of Chemical Theory and Computation*, pages 906–915, 2019. doi: 10.1021/acs.jctc.8b00959. URL https://doi.org/10.1021/acs.jctc.8b00959.

[155] Yumeng Li, Weirong Xiao, and Pingfeng Wang. Uncertainty quantification of artificial neural network based machine learning potentials. In *Volume 12: Materials: Genetics to Structures*, 11 2018. doi: 10.1115/imece2018-88071. URL https://doi.org/10.1115/imece2018-88071.

[156] V. Botu, R. Batra, J. Chapman, and R. Ramprasad. Machine learning force fields: Construction, validation, and outlook. *The Journal of Physical Chemistry C*, 121(1):511–522, 2016. doi: 10.1021/acs.jpcc.6b10908. URL https://doi.org/10.1021/acs.jpcc.6b10908.

[157] Geir K. Nilsen, Antonella Z. Munthe-Kaas, Hans J. Skaug, and Morten Brun. Epistemic uncertainty quantification in deep learning classification by the delta method. *CoRR*, 2019. URL http://arxiv.org/abs/1912.00832v2.

[158] autograd: https://github.com/HIPS/autograd. URL https://github.com/HIPS/autograd.

[159] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the Hessian in deep learning: Singularity and beyond. *CoRR*, 2016. URL http://arxiv.org/abs/1611.07476v2.

[160] Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *CoRR*, 2018. URL http://arxiv.org/abs/1812.04754v1.

[161] Jeff Gill and Gary King. What to do when your Hessian is not invertible. *Sociological Methods & Research*, 33(1):54–87, 2004. doi: 10.1177/0049124103262681. URL https://doi.org/10.1177/0049124103262681.

[162] Sheung Hun Cheng and Nicholas J. Higham. A modified cholesky algorithm based on a symmetric indefinite factorization. *SIAM J. MATRIX ANAL. APPL*, 1998.

[163] PyTorch: https://pytorch.org/. URL https://pytorch.org/.

[164] Brett W. Larsen, Stanislav Fort, Nic Becker, and Surya Ganguli. How many degrees of freedom do we need to train deep networks: A loss landscape perspective. *CoRR*, 2021. URL http://arxiv.org/abs/2107.05802v1.

[165] Salvatore Ingrassia and Isabella Morlini. Equivalent number of degrees of freedom for neural networks. In Reinhold Decker and Hans J. Lenz, editors, *Advances in Data Analysis*, pages 229–236, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-70981-7.

[166] Tianxiang Gao and Vladimir Jojic. Degrees of freedom in deep neural networks. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, pages 232–241, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115. URL https://www.auai.org/uai2016/proceedings/papers/257.pdf.

[167] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of Chemical Physics*, 134(7):074106, 2011. doi: 10.1063/1.3553717. URL https://doi.org/10.1063/1.3553717.

[168] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand. WACSF-weighted atom-centered symmetry functions as descriptors in machine learning potentials. *The Journal of Chemical Physics*, 148(24):241709, 2018. doi: 10.1063/1.5019667. URL https://doi.org/10.1063/1.5019667.

[169] Mingjie Liu and John R. Kitchin. SingleNN: Modified Behler-Parrinello neural network with shared weights for atomistic simulations with transferability. *The Journal of Physical Chemistry C*, 124(32):17811–17818, 2020. doi: 10.1021/acs.jpcc.0c04225. URL https://doi.org/10.1021/acs.jpcc.0c04225.

[170] Jacob R. Boes and John R. Kitchin. Modeling segregation on AuPd(111) surfaces with density functional theory and Monte Carlo simulations. *The Journal of Physical Chemistry C*, 121(6):3479–3487, 2017. doi: 10.1021/acs.jpcc.6b12752. URL https://doi.org/10.1021/acs.jpcc.6b12752.

[171] K. Miki, M. Panesi, E.E. Prudencio, and S. Prudhomme. Probabilistic models and uncertainty quantification for the ionization reaction rate of atomic nitrogen. *Journal of Computational Physics*, 231(9):3871–3886, 2012. doi: 10.1016/j.jcp.2012.01.005. URL https://doi.org/10.1016/j.jcp.2012.01.005.

[172] László Zimányi, Áron Sipos, Ferenc Sarlós, Rita Nagypál, and Géza I. Groma. Machine-learning model selection and parameter estimation from kinetic data of complex first-order reaction systems. *PLOS ONE*, 16(8):e0255675, 2021. doi: 10.1371/journal.pone.0255675. URL https://doi.org/10.1371/journal.pone.0255675.

[173] Nils E Napp and Ryan P Adams. Message passing inference with chemical reaction networks. *Advances in neural information processing systems*, 26:2247–2255, 2013.

[174] Alexandre Allard, Nicolas Fischer, Géraldine Ebrard, Bruno Hay, Peter Harris, Louise Wright, Denis Rochais, and Jeremie Mattout. A multi-thermogram-based Bayesian model for the determination of the thermal diffusivity of a material. *Metrologia*, 53(1):S1–S9, 2015. doi: 10.1088/0026-1394/53/1/s1. URL https://doi.org/10.1088/0026-1394/53/1/s1.

[175] Michael Griebel and Jan Hamaekers. Molecular dynamics simulations of the elastic moduli of polymer-carbon nanotube composites. *Computer Methods in Applied Mechanics and Engineering*, 193(17-20):1773–1788, 2004. doi: 10.1016/j.cma.2003.12.025. URL https://doi.org/10.1016/j.cma.2003.12.025.

[176] Yang Kang, Dunhong Zhou, Qiang Wu, Fuyan Duan, Rufang Yao, and Kun Cai. Fully atomistic molecular dynamics computation of physico-mechanical properties of PB, PS, and SBS. *Nanomaterials*, 9(8):1088, 2019. doi: 10.3390/nano9081088. URL https://doi.org/10.3390/nano9081088.

[177] Muhammad Akbar Elnanda Dzulfikar, Dyah Hikmawati, and Adri Supardi. Molecular dynamics simulation to determine elastic constant and bulk modulus from Mg_xZn. In *The 2nd International Conference On Physical Instrumentation And Advanced Materials 2019*, 2020. doi: 10.1063/5.0035227. URL https://doi.org/10.1063/5.0035227.

[178] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001. doi: 10.1111/1467-9868.00294. URL https://doi.org/10.1111/1467-9868.00294.

[179] Bratislav Lukić, Dominique Saletti, and Pascal Forquin. Use of simulated experiments for material characterization of brittle materials subjected to high strain rate dynamic tension. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375 (2085):20160168, 2017. doi: 10.1098/rsta.2016.0168. URL https://doi.org/10.1098/rsta.2016.0168.

[180] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. doi: 10.1007/s10994-021-05946-3. URL https://doi.org/10.1007/s10994-021-05946-3.

[181] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *arXiv*, 2017. URL http://arxiv.org/abs/1703.04977v2.

[182] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015. doi: 10.1038/nature14541. URL https://doi.org/10.1038/nature14541.

[183] Alim B. Alchagirov, John P. Perdew, Jonathan C. Boettger, R. C. Albers, and Carlos Fiolhais. Energy and pressure versus volume: Equations of state motivated by the stabilized jellium model. *Physical Review B*, 63(22):224115, 2001. doi: 10.1103/physrevb.63.224115. URL https://doi.org/10.1103/physrevb.63.224115.

[184] James H. Rose, John R. Smith, Francisco Guinea, and John Ferrante. Universal features of the equation of state of metals. *Physical Review B*, 29(6):2963–2969, 1984. doi: 10.1103/physrevb.29.2963. URL https://doi.org/10.1103/physrevb.29.2963.

[185] J.-P Poirier and A Tarantola. A logarithmic equation of state. *Physics of the Earth and Planetary Interiors*, 109(1-2):1–8, 1998. doi: 10.1016/s0031-9201(98)00112-5. URL https://doi.org/10.1016/s0031-9201(98)00112-5.

[186] P Vinet, J Ferrante, J R Smith, and J H Rose. A universal equation of state for solids. *Journal of Physics C: Solid State Physics*, 19(20): L467–L473, 1986. doi: 10.1088/0022-3719/19/20/001. URL https://doi.org/10.1088/0022-3719/19/20/001.

[187] Dennis Grady. Equation of state for solids. *AIP Conference Proceedings*, 1426(1):800–803, 2012. doi: 10.1063/1.3686399. URL https://aip.scitation.org/doi/abs/10.1063/1.3686399.

[188] Michel H. G. Jacobs and Harry A. J. Oonk. A new equation of state based on Grover, Getting and Kennedy's empirical relation between volume and bulk modulus. The high-pressure thermodynamics of MgO. *Physical Chemistry Chemical Physics*, 2(11):2641–2646, 2000. doi: 10.1039/a910247g. URL https://doi.org/10.1039/a910247g.

[189] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[190] Paolo Gardoni, David Trejo, Marina Vannucci, and Chandan Bhattacharjee. Probabilistic models for modulus of elasticity of self-consolidated concrete: Bayesian approach. *Journal of Engineering Mechanics*, 135(4):295–306, 2009. doi: 10.1061/(asce)0733-9399(2009)135:4(295). URL https://doi.org/10.1061/(asce)0733-9399(2009)135:4(295).

[191] W.F. Wu and C.C. Ni. Probabilistic models of fatigue crack propagation and their experimental verification. *Probabilistic Engineering Mechanics*, 19(3):247–257, 2004. doi: 10.1016/j.probengmech.2004.02.008. URL https://doi.org/10.1016/j.probengmech.2004.02.008.

[192] Ahmad BahooToroody, Filippo De Carlo, Nicola Paltrinieri, Mario Tucci, and P.H.A.J.M. Van Gelder. Bayesian regression based condition monitoring approach for effective reliability prediction of random processes in autonomous energy supply operation. *Reliability Engineering & System Safety*, 201:106966, 2020. doi: 10.1016/j.ress.2020.106966. URL https://doi.org/10.1016/j.ress.2020.106966.

[193] Yusheng Lu, Xin Peng, Dan Yang, Chao Jiang, and Weimin Zhong. The probabilistic discriminative time-series model with latent variables and its application to industrial chemical process modeling. *Chemical Engineering Journal*, 423:130298, 2021. doi: 10.1016/j.cej.2021.130298. URL https://doi.org/10.1016/j.cej.2021.130298.

[194] Tao Chen and Yue Sun. Probabilistic contribution analysis for statistical process monitoring: A missing variable approach. *Control Engineering Practice*, 17(4):469–477, 2009. doi: 10.1016/j.conengprac.2008.09.005. URL https://doi.org/10.1016/j.conengprac.2008.09.005.

[195] Jose E. Tabora, Jacob Albrecht, and Brendan Mack. *Probabilistic Models for Forecasting Process Robustness*, pages 919–935. Chemical Engineering in the Pharmaceutical Industry. John Wiley & Sons, Inc., 2019. doi: 10.1002/9781119600800.ch41. URL https://doi.org/10.1002/9781119600800.ch41.

[196] Stefano Andreon and Brian Weaver. *Bayesian Methods for the Physical Sciences*. Springer Series in Astrostatistics. Springer International Publishing, 2015. doi: 10.1007/978-3-319-15287-5. URL https://doi.org/10.1007/978-3-319-15287-5.

[197] Volker L. Deringer, Albert P. Bartók, Noam Bernstein, David M. Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021. doi: 10.1021/acs.chemrev.1c00022. URL https://doi.org/10.1021/acs.chemrev.1c00022.

[198] Albert P. Bartók and Gábor Csányi. Gaussian approximation potentials: A brief tutorial introduction. *International Journal of Quantum Chemistry*, 115(16):1051–1057, 2015. doi: 10.1002/qua.24927. URL https://doi.org/10.1002/qua.24927.

[199] Davis Unruh, Reza Vatan Meidanshahi, Stephen M. Goodnick, and Gergely T. Zimányi. Training a machine-learning driven Gaussian approximation potential for Si-H interactions. *arXiv:2106.02946*, 2021. URL http://arxiv.org/abs/2106.02946v2.

[200] Ganesh Sivaraman, Anand Narayanan Krishnamoorthy, Matthias Baur, Christian Holm, Marius Stan, Gábor Csányi, Chris Benmore, and Álvaro Vázquez-Mayagoitia. Machine-learned interatomic potentials by active learning: Amorphous and liquid hafnium dioxide. *npj Computational Materials*, 6(1):104, 2020. doi: 10.1038/s41524-020-00367-7. URL https://doi.org/10.1038/s41524-020-00367-7.

[201] Patrick Rowe, Gábor Csányi, Dario Alfè, and Angelos Michaelides. Development of a machine learning potential for graphene. *Physical Review B*, 97(5):054303, 2018. doi: 10.1103/physrevb.97.054303. URL https://doi.org/10.1103/physrevb.97.054303.

[202] Volker L. Deringer, Miguel A. Caro, and Gábor Csányi. A general-purpose machine-learning force field for bulk and nanostructured phosphorus. *Nature Communications*, 11(1):5461, 2020. doi:

10.1038/s41467-020-19168-z. URL https://doi.org/10.1038/s41467-020-19168-z.

[203] Conrad W. Rosenbrock, Konstantin Gubaev, Alexander V. Shapeev, Livia B. Pártay, Noam Bernstein, Gábor Csányi, and Gus L. W. Hart. Machine-learned interatomic potentials for alloys and alloy phase diagrams. *npj Computational Materials*, 7(1):24, 2021. doi: 10.1038/s41524-020-00477-2. URL https://doi.org/10.1038/s41524-020-00477-2.

[204] Olli-Pekka Koistinen, Vilhjálmur Ásgeirsson, Aki Vehtari, and Hannes Jónsson. Nudged elastic band calculations accelerated with Gaussian process regression based on inverse interatomic distances. *Journal of Chemical Theory and Computation*, 15(12):6738–6751, 2019. doi: 10.1021/acs.jctc.9b00692. URL https://doi.org/10.1021/acs.jctc.9b00692.

[205] José A. Garrido Torres, Paul C. Jennings, Martin H. Hansen, Jacob R. Boes, and Thomas Bligaard. Low-scaling algorithm for nudged elastic band calculations using a surrogate machine learning model. *Physical Review Letters*, 122(15):156001, 2019. doi: 10.1103/physrevlett.122.156001. URL https://doi.org/10.1103/physrevlett.122.156001.

[206] Malthe K. Bisbo and Bjørk Hammer. Efficient global structure optimization with a machine-learned surrogate model. *Physical Review Letters*, 124(8):086102, 2020. doi: 10.1103/physrevlett.124.086102. URL https://doi.org/10.1103/physrevlett.124.086102.

[207] Indranil Pan and Daya Shankar Pandey. Incorporating uncertainty in data driven regression models of fluidized bed gasification: A Bayesian approach. *Fuel Processing Technology*, 142:305–314, 2016. doi: 10.1016/j.fuproc.2015.10.027. URL https://doi.org/10.1016/j.fuproc.2015.10.027.

[208] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

[209] Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What are Bayesian neural network posteriors really like? *arXiv:2104.14421*, 2021. URL http://arxiv.org/abs/2104.14421v1.

[210] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

[211] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[212] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2013. URL http://arxiv.org/abs/1312.6114v10.

[213] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980v9*, 2014. URL http://arxiv.org/abs/1412.6980v9.

[214] Radford M. Neal. *MCMC Using Hamiltonian Dynamics*, chapter 5. CRC Press, 2011. doi: 10.1201/b10905-7.

[215] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[216] Agathe Girard. *Approximate methods for propagation of uncertainty with Gaussian process models*. University of Glasgow (United Kingdom), 2004.

[217] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[218] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[219] Jacob R. Boes, Mitchell C. Groenenboom, John A. Keith, and John R. Kitchin. Neural network and ReaxFF comparison for Au properties. *International Journal of Quantum Chemistry*, 116(13):979–987, 2016. doi: 10.1002/qua.25115. URL https://doi.org/10.1002/qua.25115.

[220] B. Mayer, H. Anton, E. Bott, M. Methfessel, J. Sticht, J. Harris, and P.C. Schmidt. Ab-initio calculation of the elastic constants and thermal expansion coefficients of Laves phases. *Intermetallics*, 11(1):23–32, 2003. doi: 10.1016/s0966-9795(02)00127-9. URL https://doi.org/10.1016/s0966-9795(02)00127-9.

[221] C. L. Fu and K. M. Ho. First-principles calculation of the equilibrium ground-state properties of transition metals: Applications to Nb and Mo. *Physical Review B*, 28(10):5480–5486, 1983. doi: 10.1103/physrevb.28.5480. URL https://doi.org/10.1103/physrevb.28.5480.

[222] M. Hebbache and M. Zemzemi. Ab initio study of high-pressure behavior of a low compressibility metal and a hard material: Osmium and diamond. *Physical Review B*, 70(22):224107, 2004. doi: 10.1103/physrevb.70.224107. URL https://doi.org/10.1103/physrevb.70.224107.

[223] Clemens Elster and Gerd Wübbeler. Bayesian regression versus application of least squares-an example. *Metrologia*, 53(1):S10–S16, 2015. doi: 10.1088/0026-1394/53/1/s10. URL https://doi.org/10.1088/0026-1394/53/1/s10.

[224] Dan Lu, Ming Ye, and Mary C. Hill. Analysis of regression confidence intervals and Bayesian credible intervals for uncertainty quantification. *Water Resources Research*, 48(9), 2012. doi: 10.1029/2011wr011289. URL https://doi.org/10.1029/2011wr011289.

[225] G J P Kok, A M H van der Veen, P M Harris, I M Smith, and C Elster. Bayesian analysis of a flow meter calibration problem. *Metrologia*, 52(2): 392–399, 2015. doi: 10.1088/0026-1394/52/2/392. URL https://doi.org/10.1088/0026-1394/52/2/392.

[226] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4405–4423, 2020. doi: 10.1109/tnnls.2019.2957109. URL https://doi.org/10.1109/tnnls.2019.2957109.

[227] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. Fast direct methods for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):252–265, 2016. doi: 10.1109/tpami.2015.2448083. URL https://doi.org/10.1109/tpami.2015.2448083.

[228] Ke Alexander Wang, Geoff Pleiss, Jacob R. Gardner, Stephen Tyree, Kilian Q. Weinberger, and Andrew Gordon Wilson. Exact Gaussian processes on a million data points. *arXiv:1903.08114*, 2019. URL http://arxiv.org/abs/1903.08114v2.

[229] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. *arXiv:1309.6835*, 2013. URL http://arxiv.org/abs/1309.6835v1.

[230] A. Banerjee, D. B. Dunson, and S. T. Tokdar. Efficient Gaussian process regression for large datasets. *Biometrika*, 100(1):75–89, 2012. doi: 10.1093/biomet/ass068. URL https://doi.org/10.1093/biomet/ass068.

[231] Samuel S. Schoenholz and Ekin D. Cubuk. Jax, M.D.: A framework for differentiable physics. *CoRR*, 2019. URL http://arxiv.org/abs/1912.04232v2.

[232] Stephan Thaler and Julija Zavadlav. Learning neural network potentials from experimental data via differentiable trajectory reweighting. *CoRR*, 2021. URL http://arxiv.org/abs/2106.01138v1.

[233] Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14):145301, 2018. doi: 10.1103/physrevlett.120.145301. URL https://doi.org/10.1103/physrevlett.120.145301.

[234] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5), 2017. doi: 10.1126/sciadv.1603015. URL https://doi.org/10.1126/sciadv.1603015.

[235] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature Communications*, 9(1):3887, 2018. doi: 10.1038/s41467-018-06169-2. URL https://doi.org/10.1038/s41467-018-06169-2.

[236] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. SchNet - A deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018. doi: 10.1063/1.5019779. URL https://doi.org/10.1063/1.5019779.

[237] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016. doi: 10.1007/s10822-016-9938-8. URL https://doi.org/10.1007/s10822-016-9938-8.

[238] Oliver T. Unke and Markus Meuwly. PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, 2019. doi: 10.1021/acs.jctc.9b00181. URL https://doi.org/10.1021/acs.jctc.9b00181.

[239] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. SE(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *CoRR*, 2021. URL http://arxiv.org/abs/2101.03164v2.

[240] Johannes Klicpera, Florian Becker, and Stephan Günnemann. GemNet: Universal directional graph neural networks for molecules. *CoRR*, 2021. URL http://arxiv.org/abs/2106.08903v2.

[241] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *CoRR*, 2020. URL http://arxiv.org/abs/2003.03123v1.

[242] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open Catalyst 2020 (OC20) dataset and community challenges. *ACS Catalysis*, 11(10):6059–6072, 2021. doi: 10.1021/acscatal.0c04525. URL https://doi.org/10.1021/acscatal.0c04525.

[243] Muhammed Shuaibi, Adeesh Kolluru, Abhishek Das, Aditya Grover, Anuroop Sriram, Zachary Ulissi, and C. Lawrence Zitnick. Rotation invariant graph neural networks using spin convolutions. *CoRR*, 2021. URL http://arxiv.org/abs/2106.09575v1.

[244] Yu Li, Xiaohong Zhang, Ashwin Srinath, Rachel B. Getman, and Linh B. Ngo. Combining HPC and big data infrastructures in large-scale post-processing of simulation data. In *Proceedings of the Practice and Experience on Advanced Research Computing*, 7 2018. doi: 10.1145/3219104.3229279. URL https://doi.org/10.1145/3219104.3229279.

[245] Douglas M. Reitz and Estela Blaisten-Barojas. Simulating the NaK eutectic alloy with Monte Carlo and machine learning. *Scientific Reports*, 9(1):704, 2019. doi: 10.1038/s41598-018-36574-y. URL https://doi.org/10.1038/s41598-018-36574-y.

[246] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. *CoRR*, 2015. URL http://arxiv.org/abs/1503.05671v7.

A Details for Ni-Al-W data

We describe some additional details about the datasets. The data can be found on https://euler.phys.cmu.edu/widom/tmp/ForJK/ in vasprun.xml files, and specific paths and the db filenames are in Table A.1. Datasets A, B, D, G, H have additional subfolders following the listed paths. For Datasets A, B, C, D, we used the vasprun.xml files only. Datasets E, G, H, J have vasprun.xml.mdxxxx files which are continuous MD trajectories in between attempted MC steps. For Dataset D, 96/../series1/ were not used because they were for DFT testing. We did not use Datasets H and J, but they could be used in future work. In Dataset J, the volumes range from 12.0 to 14.5 $Å^3$/atom, and the Vrun file gives the volume within the T=* directory. Dataset F was not from the website but from an outcar.1 file.

Table A.1: Datasets with paths

	Path	db file
A.	32/long-runs/	ni-al-w-def.db
B.	500/	ni-al-w-500.db
C.	64/series1/2000/, ..., 64/series7/2000/, 64/series2/2000/convergence/	ni-al-w-64.db
D.	96/V+1/series2/, 96/V-1/series2/, 96/V/series2/	ni-al-w-96-2-s2.db
E.	96-more/V+1/, 96-more/V-1/, 96-more/V/Al14Ni77W5/, ..., 96-more/V/Al12Ni79W5/	ni-al-w-96-2200-v+1.db, ni-al-w-96-2200-v-1.db, ni-al-w-96-2200-al14ni77w5.db, ni-al-w-96-2200-al12ni79w5.db, ni-al-w-96-2200-al16ni77w3.db, ni-al-w-96-2200-al17ni73w6.db, ni-al-w-96-2200-all.db (former 6 db consolidated)
F.		outcar.1, outcar-500.db
G.	32-more/	ni-al-w-32-more-all.db
H.	N96-take3/V+1/, N96-take3/V-1/, N96-take3/V/	
J.	replica-more/T=1720/, ..., replica-more/T=2470/	

B Settings for hyperparameter study

Table B.1: Number of radial functions detailed settings

No. of sets G^2	η's
1	0.25
2	0.08, 0.25
3	0.03, 0.08, 0.25
4	0, 0.03, 0.08, 0.25
5	0, 0.02, 0.06, 0.18, 0.4
6	0.012, 0.025, 0.05, 0.10, 0.23, 0.50
9	0, 0.01, 0.02, 0.035, 0.055, 0.08, 0.13, 0.23, 0.4
10	0, 0.01, 0.02, 0.035, 0.055, 0.08, 0.13, 0.23, 0.4, 0.75

Table B.2: Shifted and non-shifted radial functions detailed settings

No.	Model	η	R_s
1	4shift	0.4	0
		0.4	3.1
		0.4	6.4
		0.4	9.1
2	1eta-3shift	0.03	0
		0.4	3.5
		0.4	7
		0.4	9.8
3	2eta-2shift	0.01	0
		0.08	0
		0.4	3.5
		0.4	7
4	2eta-2shift	0.2	0
		0.22	0
		0.4	1.5
		0.4	7
5	3eta-1shift	0	0
		0.03	0
		0.25	0
		0.4	5.5
6	3eta-1shift	0.04	0
		0.09	0
		0.28	0
		0.4	6.5
7	4eta	0.04	0
		0.08	0
		0.25	0
		0.6	0

Table B.3: Cutoff radius detailed settings

Cutoff radius (Bohr)	η's
8	0.02, 0.06, 0.13, 0.25
9	0.02, 0.06, 0.13, 0.25
12	0, 0.03, 0.08, 0.25
14	0.02, 0.06, 0.13, 0.3

Table B.4: Angular functions and four radial functions detailed settings

No.	Model	η's	λ	ζ
1	1ang	0.02, 0.05, 0.10, 0.35		
			-1	1
2	1ang	0.02, 0.05, 0.10, 0.35		
			1	2
3	1ang	0.02, 0.05, 0.10, 0.35		
			-1	2
4	1ang	0.02, 0.05, 0.10, 0.35		
			1	4
5	2ang	0.03, 0.06, 0.13, 0.40		
			1	8
			1	12
6	2ang	0.03, 0.06, 0.13, 0.40		
			1	12
			1	24
7	2ang	0.02, 0.05, 0.10, 0.35		
			1	1
			1	4
8	4ang	0.02, 0.05, 0.10, 0.35		
			1	2
			1	4
			1	8
			1	16
9	4ang	0.02, 0.05, 0.10, 0.35		
			-1	1
			1	2
			1	4
			1	8

Table B.5: Additional angular models with variable number of radial functions

No.	Model	Val. Energy RMSE	Val. Force RMSE	Time (hrs)	Epochs
1	1ang-3rad	0.00815	0.3283	3.4	85
2	1ang-3rad	0.00109	0.0340	2.1	53
3	1ang-2rad	0.00344	0.0971	3.3	83
4	1ang-3rad	0.00031	0.0097	4.6	100
5	2ang-2rad	0.00064	0.0199	7.1	100
6	2ang-2rad	0.00121	0.0532	7.9	100

Table B.6: Additional angular and radial models detailed settings

No.	Model	η's	λ	ζ
1	1ang-3rad	0.03, 0.08, 0.25		
		0.03	1	12
2	1ang-3rad	0.03, 0.08, 0.25		
		0	-1	3
3	1ang-2rad	0.02, 0.22		
		0	1	12
4	1ang-3rad	0.02, 0.08, 0.22		
		0	1	8
5	2ang-2rad	0.02, 0.22		
		0	1	4
		0	1	24
6	2ang-2rad	0.02, 0.22		
		0	-1	1
		0	-1	2

C Additional MD results for Ni-Al-W

In Section 2.3.3, we included the Ni-W partial radial distribution function and Green-Kubo and Einstein diffusion constants for Al. Here we include the remaining results from the NVT simulation.

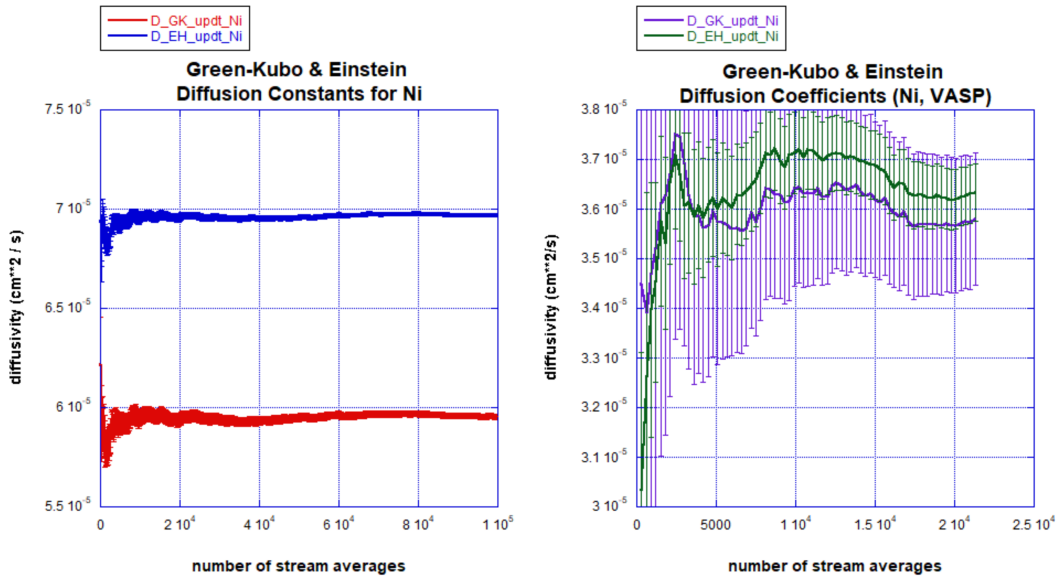


Figure C.1: Ni diffusivity for ML potential (left) and AIMD (right).

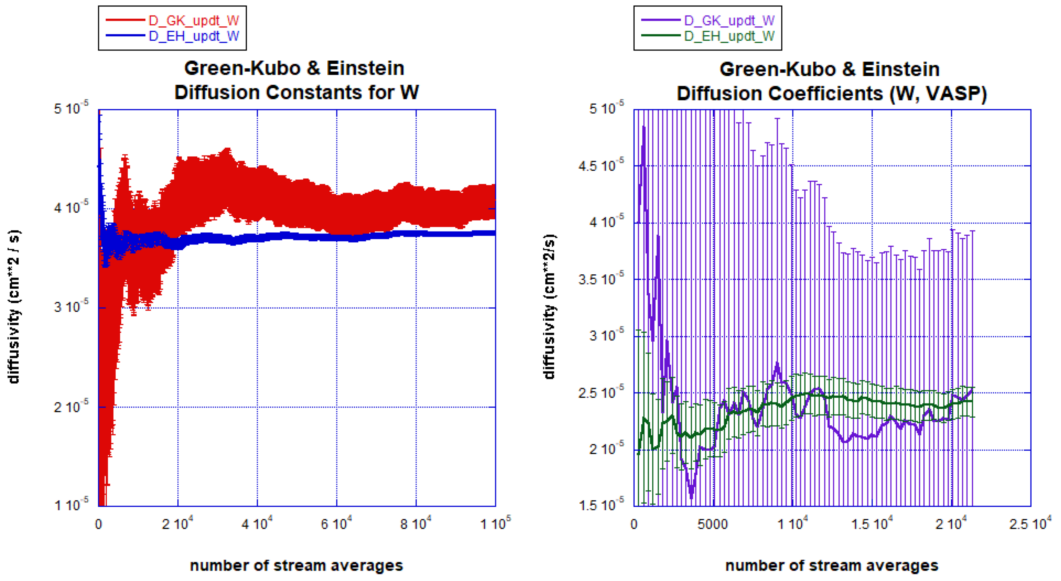


Figure C.2: W diffusivity for ML potential (left) and AIMD (right).

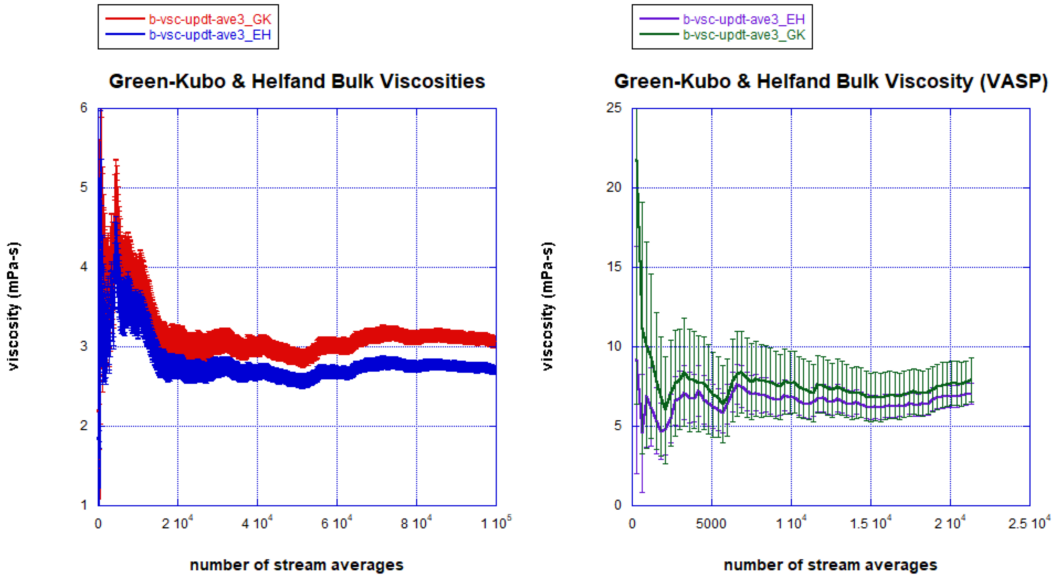


Figure C.3: Bulk viscosity for ML potential (left) and AIMD (right).

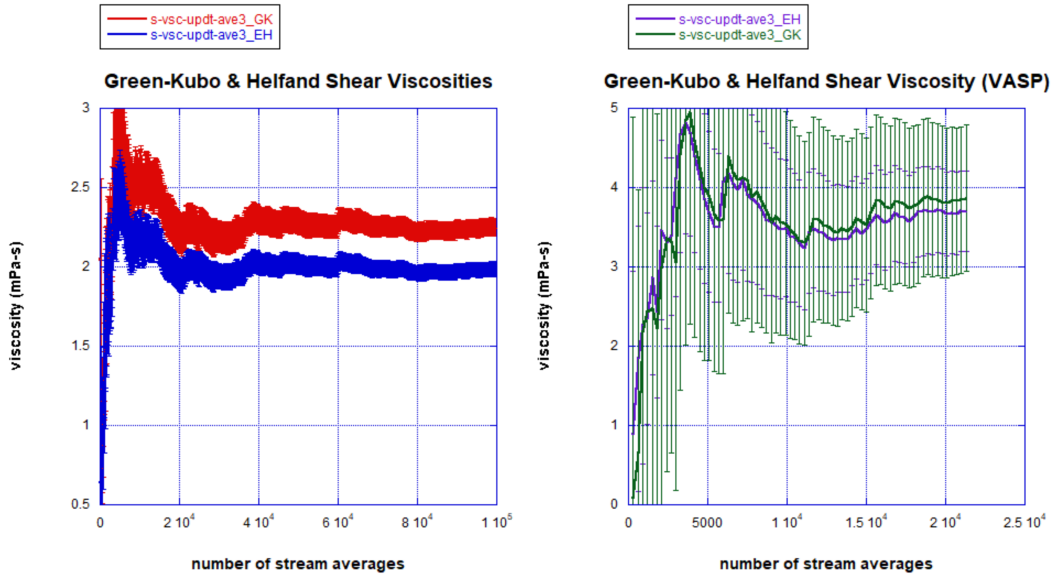


Figure C.4: Shear viscosity for ML potential (left) and AIMD (right).

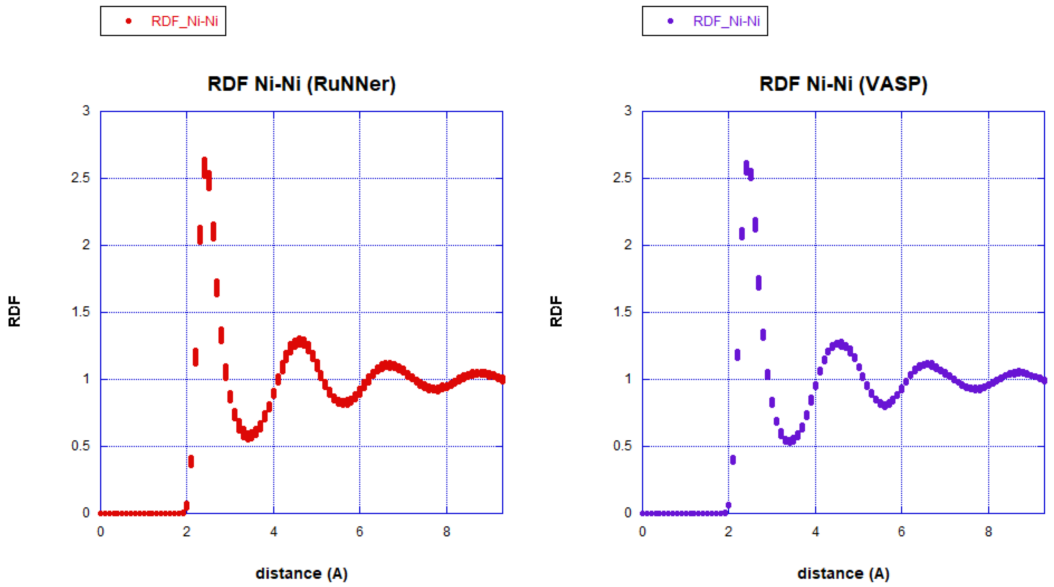


Figure C.5: Ni-Ni radial distribution function for ML potential (left) and AIMD (right).

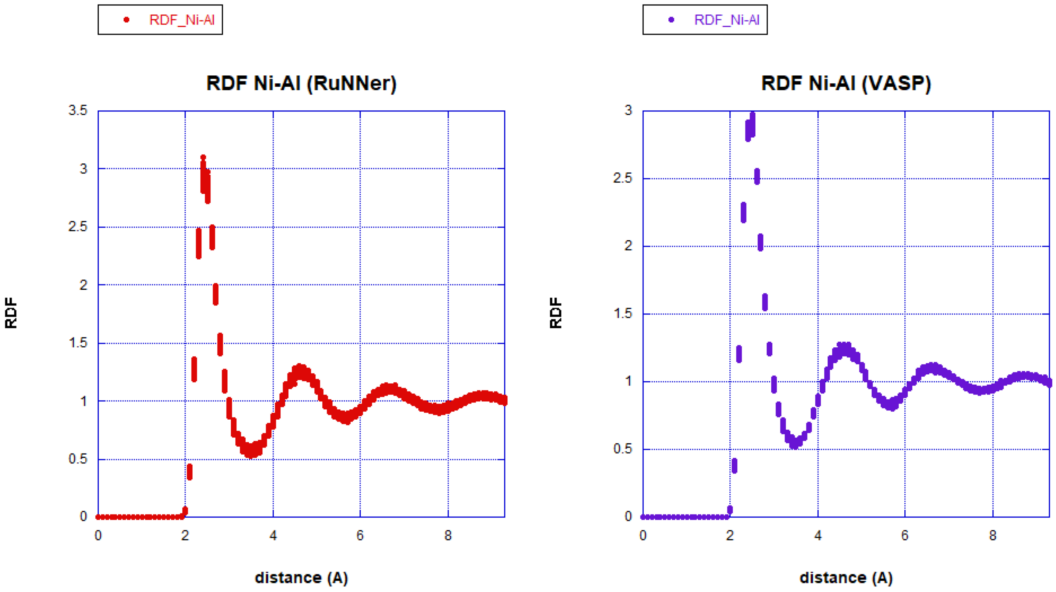


Figure C.6: Ni-Al radial distribution function for ML potential (left) and AIMD (right).

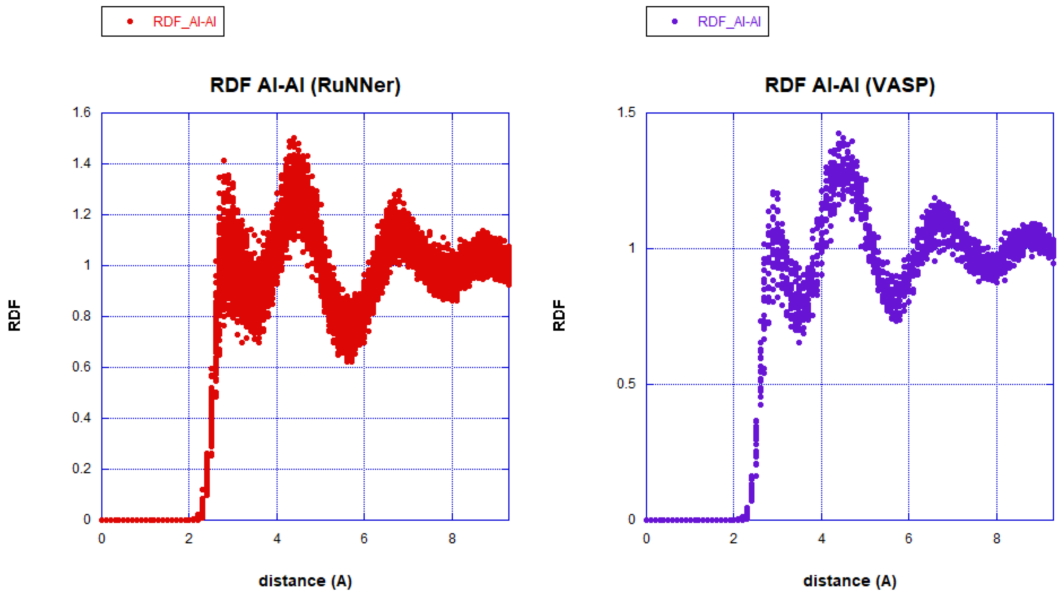


Figure C.7: Al-Al radial distribution function for ML potential (left) and AIMD (right).

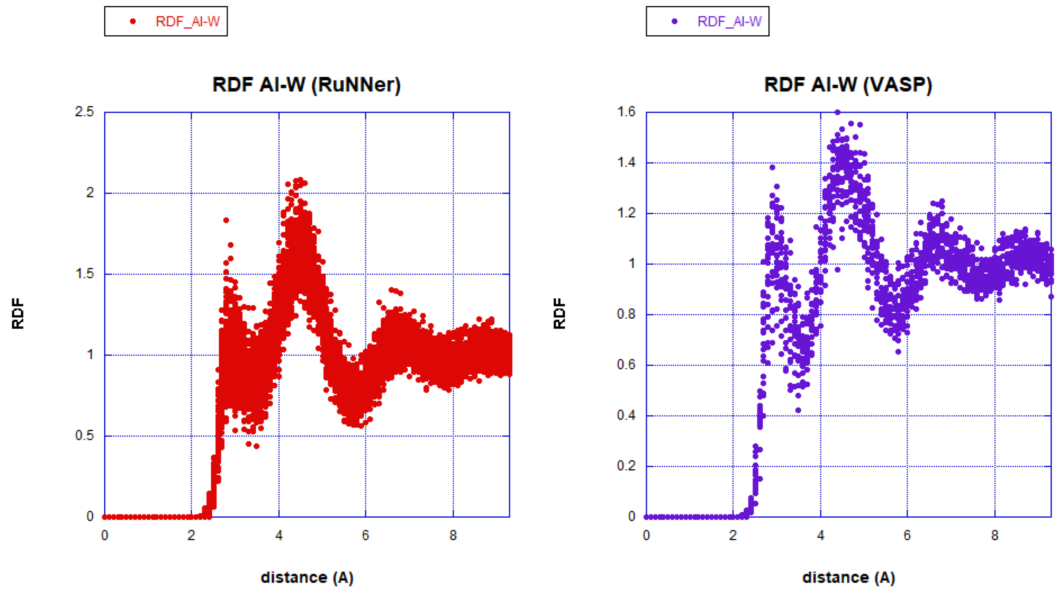


Figure C.8: Al-W radial distribution function for ML potential (left) and AIMD (right).

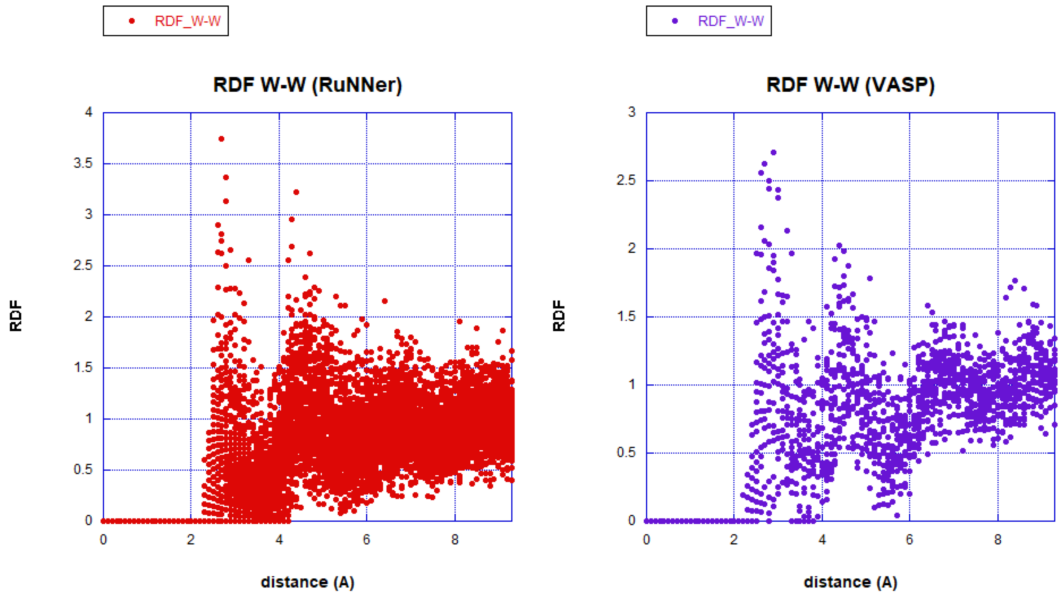


Figure C.9: W-W radial distribution function for ML potential (left) and AIMD (right).

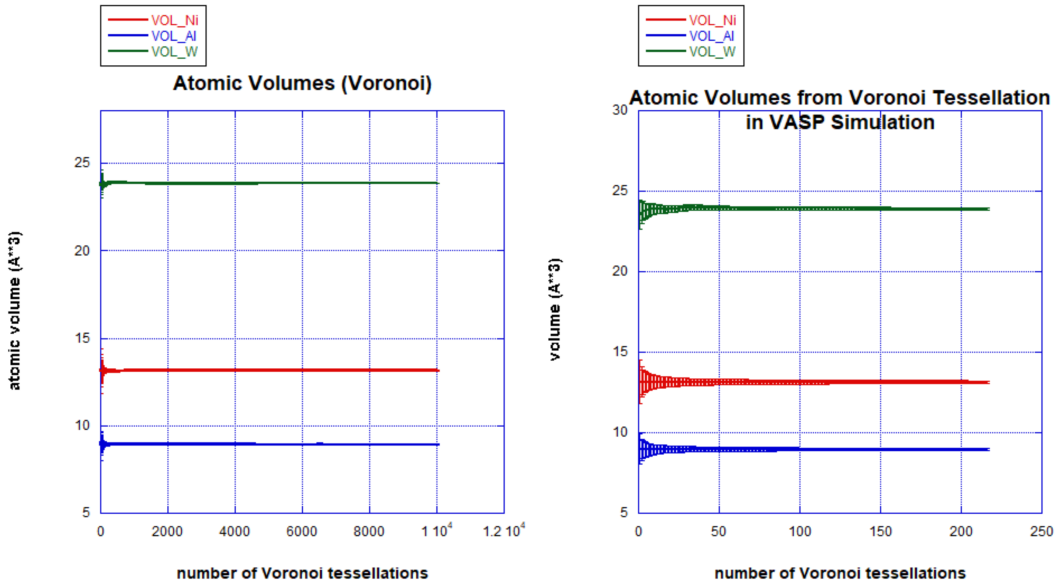


Figure C.10: Atomic volume from Voronoi tessellation for ML potential (left) and AIMD (right).

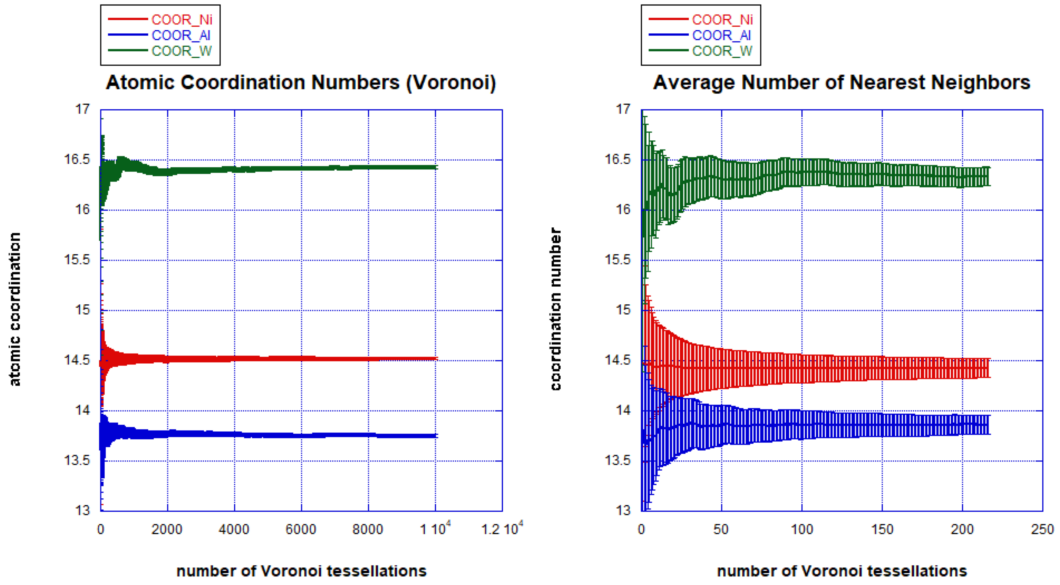


Figure C.11: Average coordination number from Voronoi tessellation for ML potential (left) and AIMD (right).

D Delta method theory

The delta method is based on regression, and gives a standard error of prediction by linearly approximating the model. We are doing a regression with data $\{x_i, y_i\}$. Our model predicts $y\left(x_i \mid \theta\right)$, and the theory of the delta method assumes that the data output is the sum of the model prediction and some Gaussian error

$$y_i = y\left(x_i \mid \theta\right) + \epsilon_i$$

with $\epsilon_i \sim N(0, \sigma_i)$, y_i as data output, x_i as data input, and θ as model parameters.

The log likelihood of the data given the model, l_n, is

$$l_n = \log P(\{y_i\} \mid \theta).$$

Since we assumed ϵ_i was Gaussian,

$$l_n \propto -\frac{1}{2} \sum_i \left(\frac{y_i - y(x_i \mid \theta)}{\sigma_i} \right)^2.$$

The above term includes the sum of squared errors which is common as the loss or regression objective function during training. In least squares regression, we minimize the sum squared errors to get the maximum likelihood estimate of parameters, $\hat{\theta}$.

The standard error of $\hat{\theta}$

$$\text{se}(\hat{\theta}) \approx \frac{1}{\sqrt{I_n(\theta)}}$$

where $I_n(\theta)$ is the Fisher information matrix defined as

$$I_n(\theta) = -\mathbb{E}_\theta \left[\frac{\partial^2 \, l_n(\{y_i\} \mid \theta)}{\partial \theta^2} \right].$$

The standard error of $\hat{\theta}$ is obtained from a Taylor's series expansion around $l_n'(\theta)$.[116] We are able to obtain this standard error by assuming $\hat{\theta}$ is centered and Gaussian around the true parameters θ.

In the Fisher information, note that l_n is the same log likelihood defined earlier, so the Fisher information is proportional to the Hessian of the loss with respect to model parameters, and thus can be readily obtained.

Now we will obtain the standard error of model prediction. Suppose for function $g(\hat{\theta})$, $g'(\hat{\theta})$ is nonzero, then

$$\text{se}(g(\hat{\theta})) \approx \sqrt{(g')^T I_n^{-1} g'}.$$

The standard error of $g(\hat{\theta})$ is obtained from a Taylor's series around $g(\theta)$ and using $\text{se}(\hat{\theta})$ obtained previously.[116]

The standard error depends on the training data because the Fisher information depends on the training data. The standard error also depends on the model, its parameters, and the point we are predicting, because these determine g'.

In this work, we assume the error ϵ_i is independent of the data point x_i. This allows the simplification

$$l_n \propto -\frac{1}{2}\sum_i \left(\frac{y_i - y(x_i \mid \theta)}{\sigma_i}\right)^2 = -\frac{1}{2\sigma^2}\sum_i \left(y_i - y(x_i \mid \theta)\right)^2.$$

We estimate σ^2 as

$$\sigma^2 \approx \frac{1}{n}\sum_i^n \left(y_i - y(x_i \mid \theta)\right)^2.$$

Once obtaining standard errors for a prediction, we can construct confidence intervals. We use $t_{\frac{\alpha}{2}} \cdot \text{se}(g(\hat{\theta}))$ for $(1-\alpha)\%$ confidence intervals. The confidence interval indicates confidence of fit. The prediction standard error has an additional term

$$\text{prediction se}(g(\hat{\theta})) = \sqrt{(g')^T I_n^{-1} g' + \sigma_r^2}$$

where σ_r^2 is residual variance and approximated by

$$\sigma_r^2 \approx \frac{1}{n}\sum_i^n \left(g_i - g(x_i \mid \theta)\right)^2.$$

A $(1-\alpha)\%$ prediction interval is then $t_{\frac{\alpha}{2}} \cdot (\text{pred. se}(g(\hat{\theta})))$. The prediction interval represents how often a new point would fall in the interval.

E Additional results for probabilistic EOS

In Section 5.3.3, we included the comparison of HMC, SVI, and delta method for Pd Poirier-Tarantola. Here we include the remaining results for Pd Birch, Au Murnaghan, and Au Vinet.

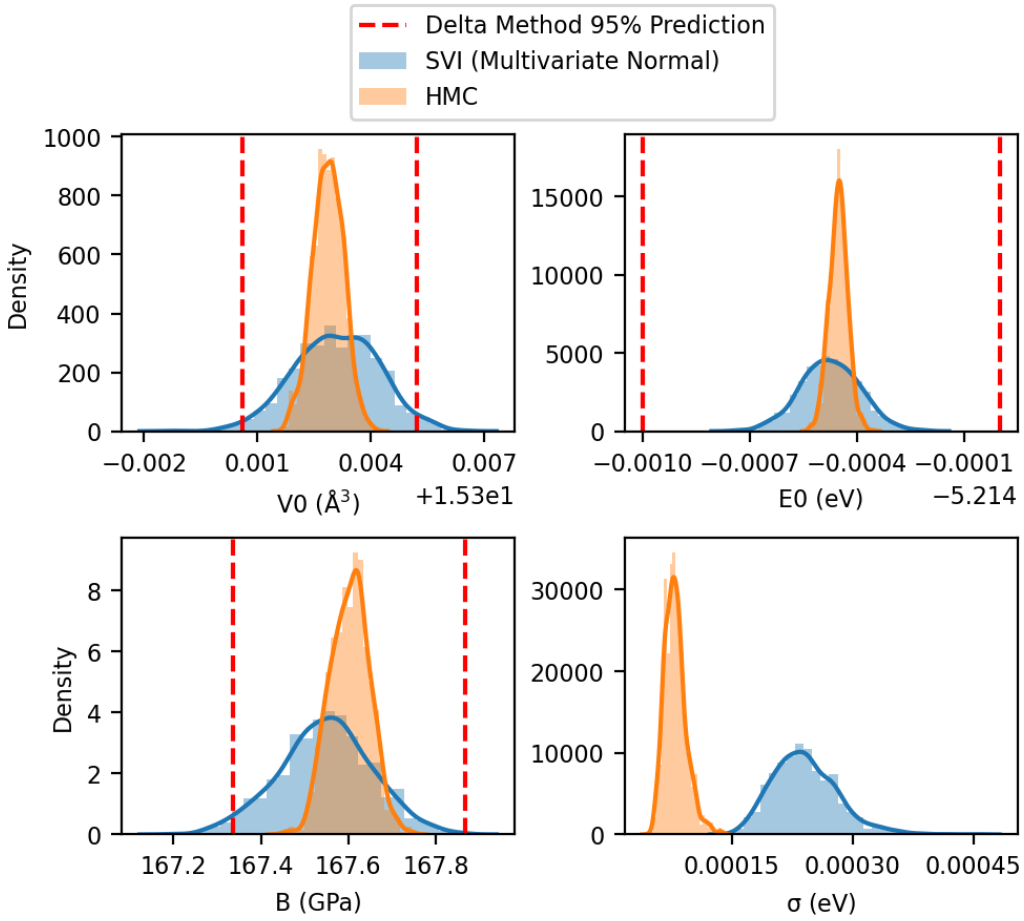


Figure E.1: Pd Birch comparison of HMC, SVI posteriors and delta method prediction interval.

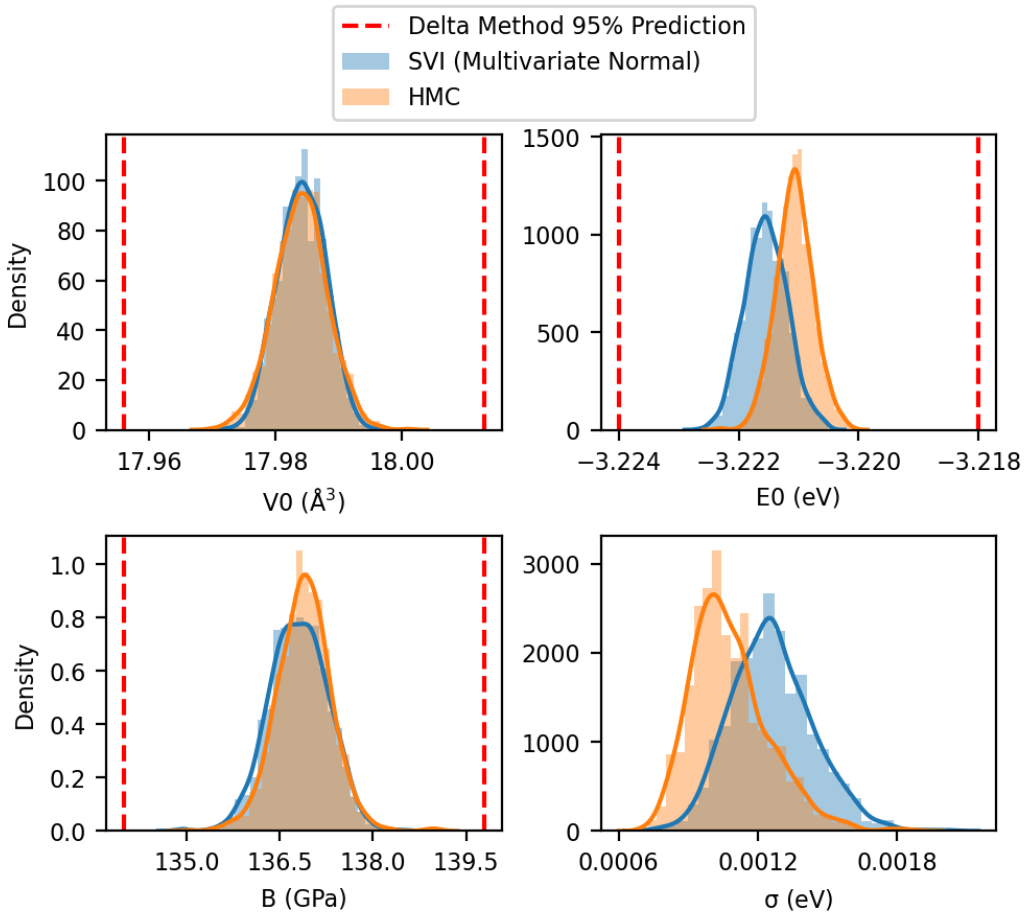


Figure E.2: Au Murnaghan comparison of HMC, SVI posteriors and delta method prediction interval.

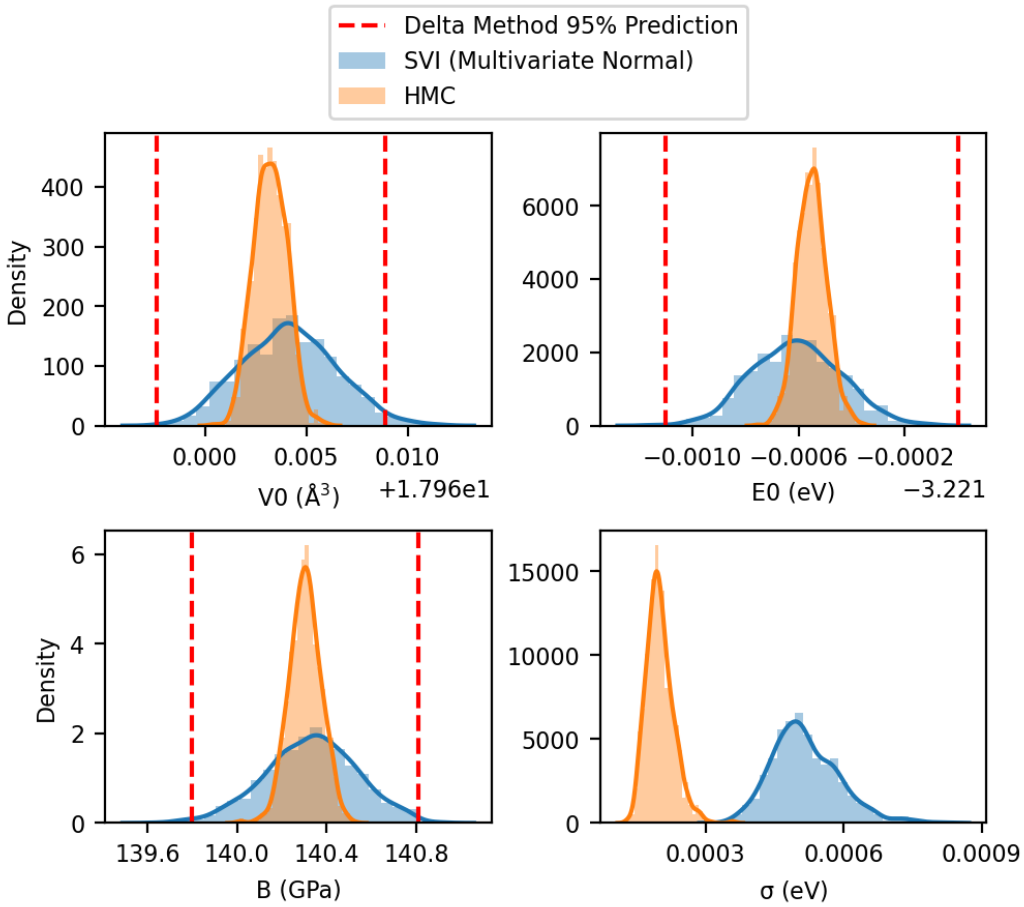


Figure E.3: Au Vinet comparison of HMC, SVI posteriors and delta method prediction interval.

Similar to results from Section 5.3.4, we include GP results for Au here.

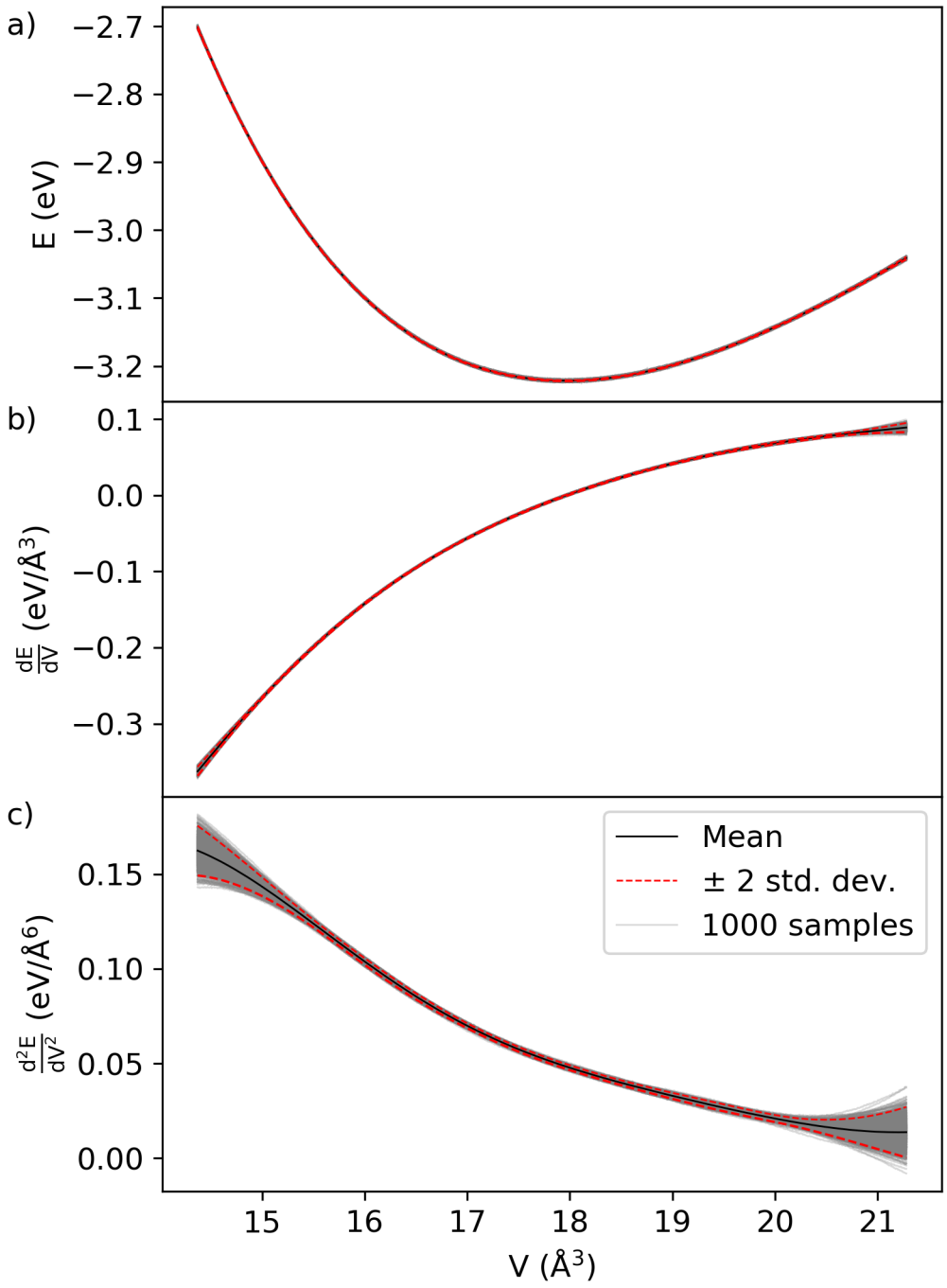


Figure E.4: Au Gaussian process posterior.

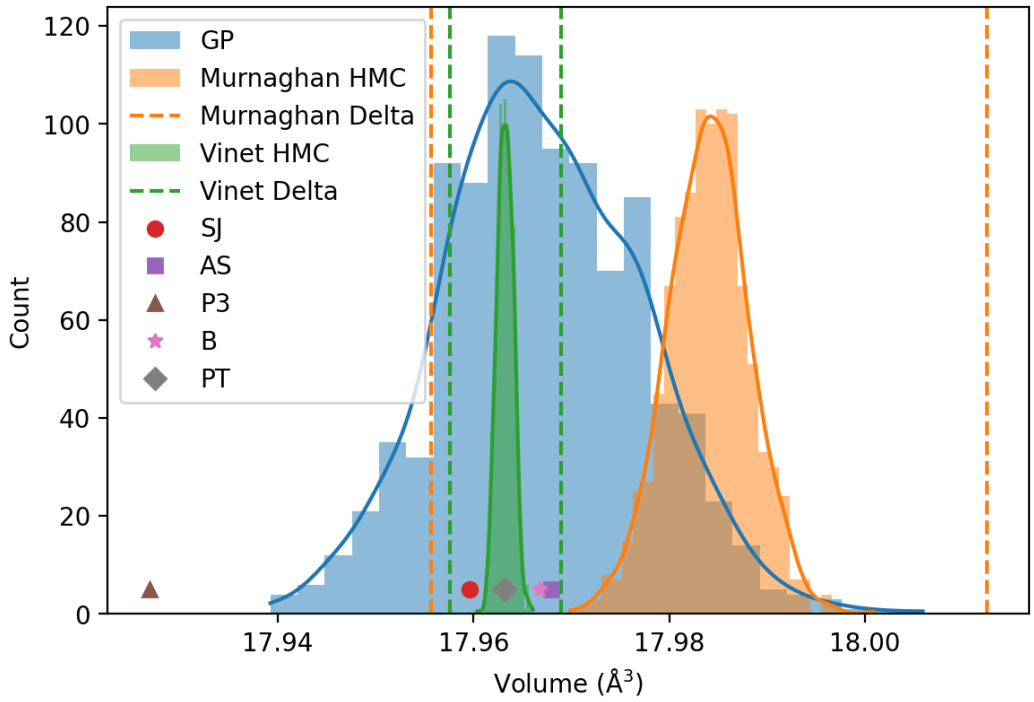


Figure E.5: Au minimum volume comparison of GP, Bayesian regression, nonlinear regression uncertainties, and different model predictions.

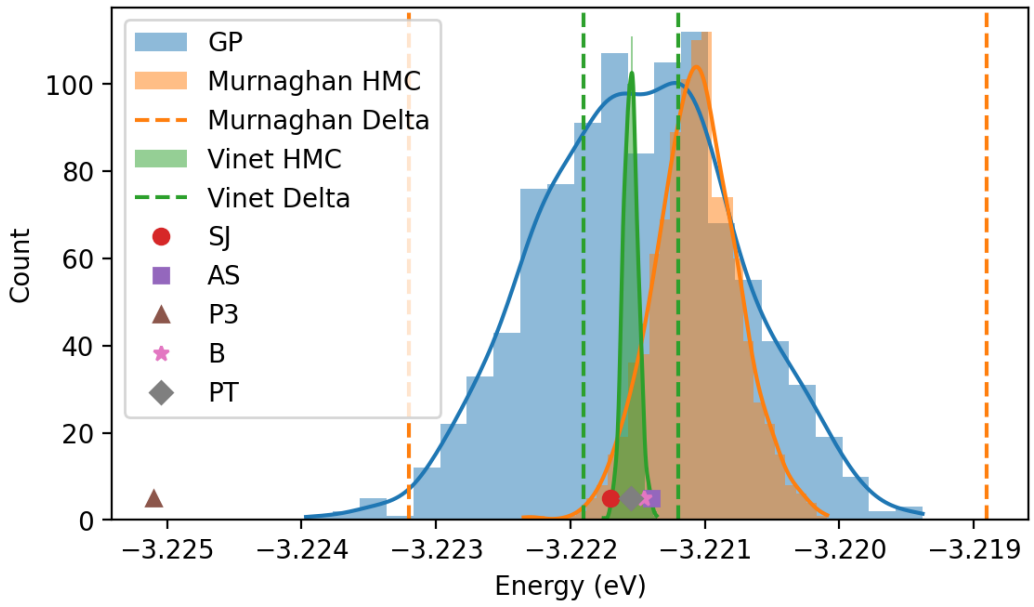


Figure E.6: Au minimum energy comparison of GP, Bayesian regression, nonlinear regression uncertainties, and different model predictions.

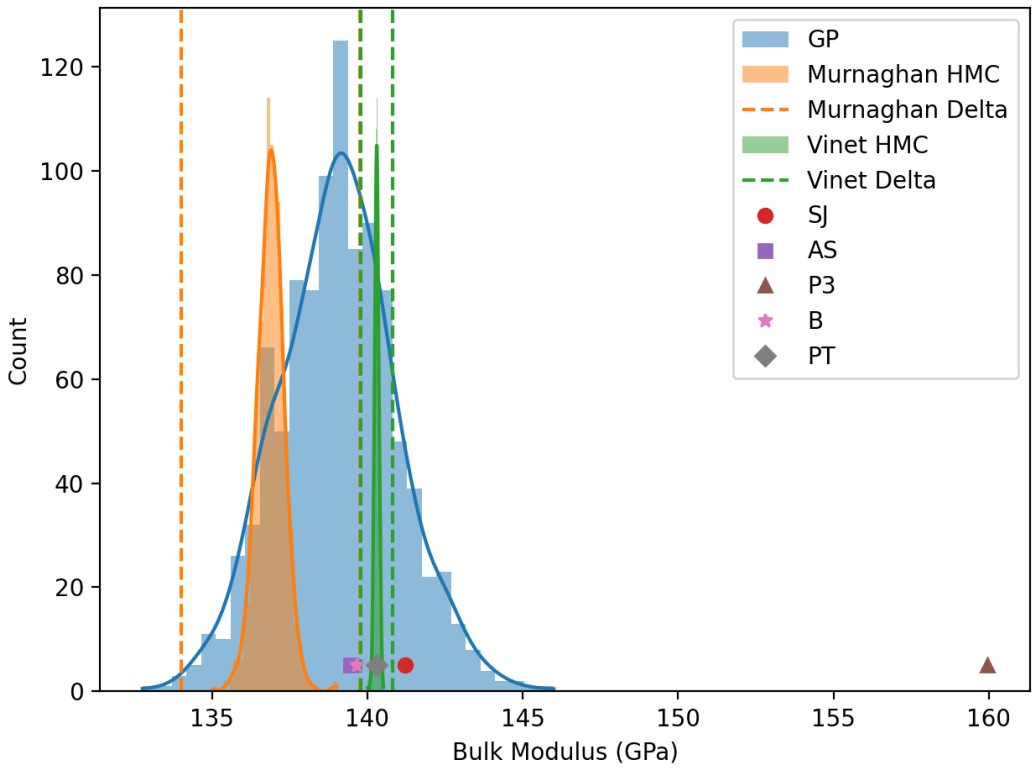


Figure E.7: Au bulk modulus comparison of GP, Bayesian regression, nonlinear regression uncertainties, and different model predictions.