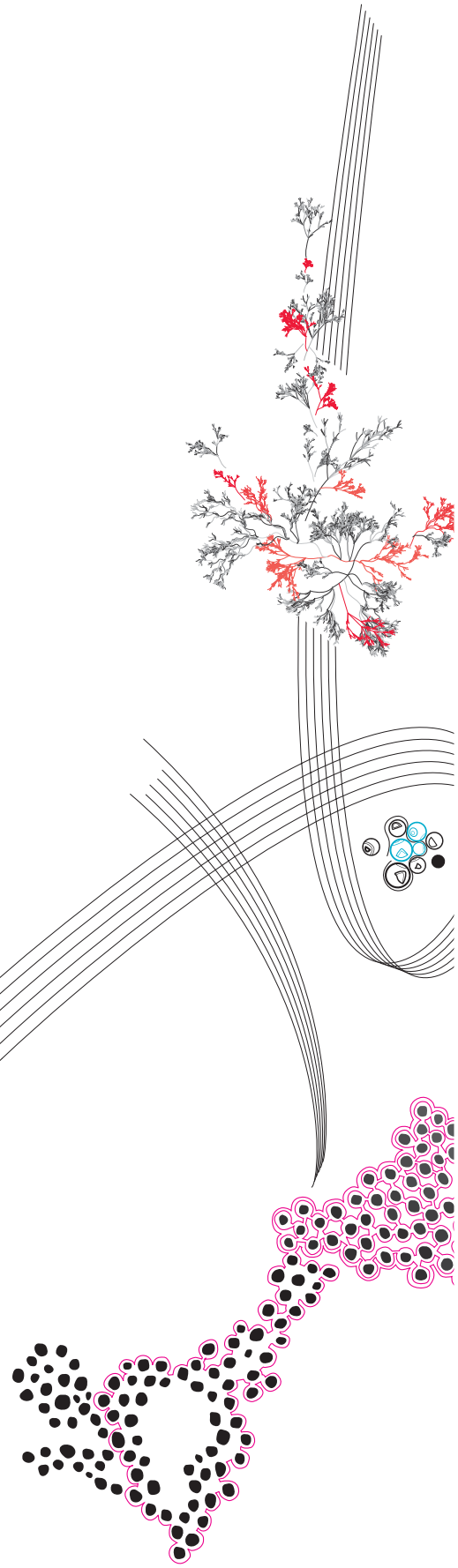


BSc Thesis Applied Mathematics and Applied Physics

Superionic Conduction simulated by Neural Network Potentials trained on On-The-Fly Force Fields

Thijmen Kuipers

Supervisor: M. Bokdam, B. J. Geurts

July, 2022

Department of Applied Physics
Faculty of Science and Technology

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I would like to thank Menno Bokdam en Bernard J. Geurts for their assistance and insightful comments during the course of this assignment. I also thank Jonathan Lahnsteiner for his help in setting up the initial neural network potential training in *RuNNer*.

Superionic Conduction simulated by Neural Network Potentials trained on On-The-Fly Force Fields

T. P. W. Kuipers

July, 2022

Abstract

We present a method to train High-Dimensional Neural Network Potentials on small ($\sim 10^2$ structures) training sets with significantly reduced computation time. This is achieved using On-The-Fly Machine Learning Force Fields for data set generation. This method is analysed on diamond, a simple solid well-described by harmonic lattice dynamics, and lithium nitride, both below and above its superionic phase transition, exhibiting difficult-to-capture anharmonic lattice dynamics. The High-Dimensional Neural Network Potential is shown to work well for diamond, but fails to capture lithium diffusion in lithium nitride well enough to perform molecular dynamics above the phase transition. We conclude with some promising improvements that might yet lead to a correct description of superionic lithium nitride with a significantly reduced training set.

Keywords: Molecular dynamics, machine learning, neural network potential, kernel ridge regression, gaussian overlap potential, superionic conduction

Contents

1 Introduction

Over the course of the last decade, the application of machine learning in molecular dynamics has seen a large increase in both efficiency and capability [1–6]. These techniques allow for very fast atomic structure calculations by training on first-principles-based (FP) electronic structure calculations.

The field of molecular dynamics (MD) concerns the nanoscopic simulation of classical motion of atoms or molecules in a material. These trajectories, obtained through numerical integration of Newtonian equations of motion, can then be used to calculate macroscopic material properties like diffusivity, heat capacity, or phase transition temperatures. The challenge is the calculation of atomic forces, which, before the advent of machine learning, could either be obtained through expensive first-principles-based electron structure calculations like density-functional theory, or inaccurate (semi-)empirical force fields. Machine learning has formed a bridge between these two extremes, allowing for both cheap and accurate first-principles-based force fields.

In particular, the application of artificial neural networks has found much success in the form of the High-Dimensional Neural Network Potential (HDNNP), originally proposed by Behler and Parrinello in 2007 [1]. The core of this technique is a simple feedforward neural network.

The training of the neural network typically requires a significant number of training structures to sufficiently cover the phase space, regularly ranging in the thousands to tens-of-thousands [2]. This becomes especially apparent when seeking a neural network potential applicable to a wide temperature range, where phase transitions may lead to the exploration of unique and sparsely sampled sections of the phase space. This forces restrictions on the simulation sizes and temperature ranges that can be covered by the neural network potential, and will often take days to weeks even with these restrictions in place.

Not all machine-learned force fields suffer the faith of lengthy training efforts. Kernel ridge regression methods allow for very fast learning, and combined with error statistics, can be applied during the generation of the training set, only performing new electronic structure calculations when predicted errors exceed some limit. Specifically, such an on-the-fly machine learning method proposed by Bokdam and co-workers [7], a derivative of the Gaussian Approximation Potential [8], has been shown to successfully remove up to 99% of the computationally expensive electronic structure calculations from the training process. This allows for rapid exploration of the phase space while keeping training set size and computation time to a minimum.

We will apply this on-the-fly method in the generation of the training set for an HDNNP. We seek to analyse its ability to train using only those structures chosen on-the-fly for electronic structure calculations. Furthermore, we propose a method of augmenting the training set when the potential energy surface obtained by the HDNNP is not satisfactory. This would reduce time needed for training set generation by orders of magnitude, allowing for a much larger part of the phase space to be represented in the HDNNP.

We will start by presenting background material and the basic structure of the HDNNP, as well as the procedure of data set augmentation in Section 2. The HDNNP will be trained on on-the-fly runs of two materials, diamond and lithium nitride. The former is a simple solid well-described by harmonic lattice dynamics [9], while the latter exhibits an anharmonic superionic phase transition [10]. These materials will show both the potential and limitations of the on-the-fly training set generation. The exact procedure and settings followed in this thesis will be outlined briefly in Section 3, followed by a discussion of the

results for diamond and lithium nitride in Section 4.

2 Theory

2.1 Molecular dynamics and potential energy regression

Molecular dynamics is the simulation of the movement of molecules or, in this case, atoms. A structure, containing a number of atoms with positions $\{\mathbf{R}_i\}$, is propagated through time by numerical integration of the Newtonian equations of motion, usually by means of the Verlet integrator. Forces can either be obtained directly from the coordinates or, more common amongst empirical methods, as gradients on the total potential energy of the structure, where $\mathbf{F}_i = -\nabla_i U(\{\mathbf{R}_j\})$. The function $U(\{\mathbf{R}_i\})$ is called the potential energy surface (PES), and allows for a convenient single-valued quantity to predict the forces.

First-principles-based methods are available to obtain both the total potential energy and atomic forces in a structure. These electronic structure calculations are expensive but accurate. We will be using density-functional theory (DFT), which will be used as ground-truth, but will not be covered in detail (exact settings were provided by the supervisor). Note, when referring to a "structure" as part of a data set, we will usually be referring to all quantities of interest for that structure: the atomic coordinates $\{\mathbf{R}_i\}$, the atomic species, the atomic force $\{\mathbf{F}_i\}$, and the total potential energy U.

There are various techniques available for regression of the PES. For a generally applicable technique, we require the PES to admit a variable number of atoms. Other physically motivated requirements are translational and rotational invariance with respect to atomic coordinates, as well as invariance to the permutation of chemically equivalent atoms (which, in this case, means atoms of the same element) [2]. In both methods of interest to this thesis, this is achieved by taking the potential energy to be the sum of local atomic contributions U_i, influenced only by atoms less than some cut-off radius r_c away,

$$U = \sum_i U_i(\{\mathbf{R}_j | r_{ij} < r_c\}), \quad r_{ij} = |\mathbf{R}_i - \mathbf{R}_j|. \tag{1}$$

Although these contributions have no physical analogue, this assumption partly resolves permutational invariance and the dependence on the number of atoms. Of course, atoms may still drift in and out of each other's cut-off radii during a molecular dynamics simulation, so one additional processing step is required.

We introduce the concept of a descriptor (also *atomic fingerprint*) for atom i,

$$\mathbf{G}_i(\{\mathbf{R}_j | r_{ij} < r_c\}), \tag{2}$$

consisting of a number of symmetry functions. These must themselves satisfy translational, rotational, and permutational invariance and serve as inputs to the regression model $U_i(\mathbf{G}_i)$ instead of the atomic coordinates. This resolves the invariances for the PES. Great care must be taken in the construction of these descriptors; they are necessarily a lossy encoding of the atomic coordinates (the number of symmetry functions is fixed, but the number of atoms inside the cut-off radius is not), but must encode enough information for the regression model to differentiate between important structures in the PES.

The construction of descriptors is generally an arduous task of trial-and-error, for which one must simply gain an intuition [2]. Recently, more automated methods of descriptor selection have been implemented, ranging from statistical analysis of descriptors seen in the training set [11], to genetic algorithms that refine descriptors based on actual training performance [12, 13]. These are not implemented in the *n2p2* code used here, so will not be used.

2.1.1 Behler-Parrinello Symmetry Functions

The Behler-Parrinello Symmetry Functions (BPSF) are a type of symmetry function proposed along with the original Neural Network Potential [1, 14] and still in common use. They are commonly referred to by their types and are element-specific. That is, each radial symmetry function must have specified what element the central atom should be, and over what element it should iterate. This way, each element combination gets its own set of symmetry functions. For angular symmetry functions, two elements need to be specified, over which the sum should iterate. The original three BPSFs are the type 2 (radial), type 3 (narrow angular), and type 9 (wide angular):

$$G_i^2 = \sum_{j \neq i} e^{-\mu(r_{ij}-r_s)^2} f_c(r_{ij}) \tag{3}$$

$$G_i^3 = 2^{1-\zeta} \sum_{\substack{j,k \neq i \\ j<k}} (1 + \lambda \cos\theta_{ijk})^\zeta \, e^{-\mu(r_{ij}-r_s)^2} e^{-\mu(r_{ik}-r_s)^2} e^{-\mu(r_{jk}-r_s)^2} f_c(r_{ij}) f_c(r_{ik}) f_c(r_{jk}) \tag{4}$$

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

$$f_c(r) = \begin{cases} \frac{1}{2}\left(1 + \cos \pi x\right), & r \leq r_c \\ 0, & r > r_c \end{cases}. \tag{6}$$

where r_c is the previously discussed cut-off radius and ζ, μ, and r_s are parameters to be tuned by the user. We refer to the *n2p2* code documentation [15] for a more complete list of available cut-off functions.

Later modifications drastically reduce the number of symmetry functions to be specified, by including an element-specific weight factor in the summation, instead of having a separate symmetry function to sum over each element [12]. Another modification is the use of piecewise polynomials instead of exponentials [13], which both speeds up computation time and completely avoids the use of a cut-off function (these symmetry functions go to zero outside their compact support on their own). The combination of the two has been shown to achieve the same accuracy as the classic BPSFs with fewer symmetry functions [13]. The combination gives the type 23 (radial), type 24 (narrow angular), and type 25 (wide angular):

$$G_i^{23} = \sum_{j \neq i} Z_j C^a(r_{ij}, r_l, r_c) \tag{7}$$

$$G_i^{24} = \sum_{\substack{j,k \neq i \\ j<k}} Z_j Z_k C^a(r_{ij}, r_l, r_c) C^a(r_{ik}, r_l, r_c) C^a(r_{jk}, r_l, r_c) C(\theta_{ijk}, \theta_l, \theta_r) \tag{8}$$

$$G_i^{25} = \sum_{\substack{j,k \neq i \\ j<k}} Z_j Z_k C^a(r_{ij}, r_l, r_c) C^a(r_{ik}, r_l, r_c) C(\theta_{ijk}, \theta_l, \theta_r) \tag{9}$$

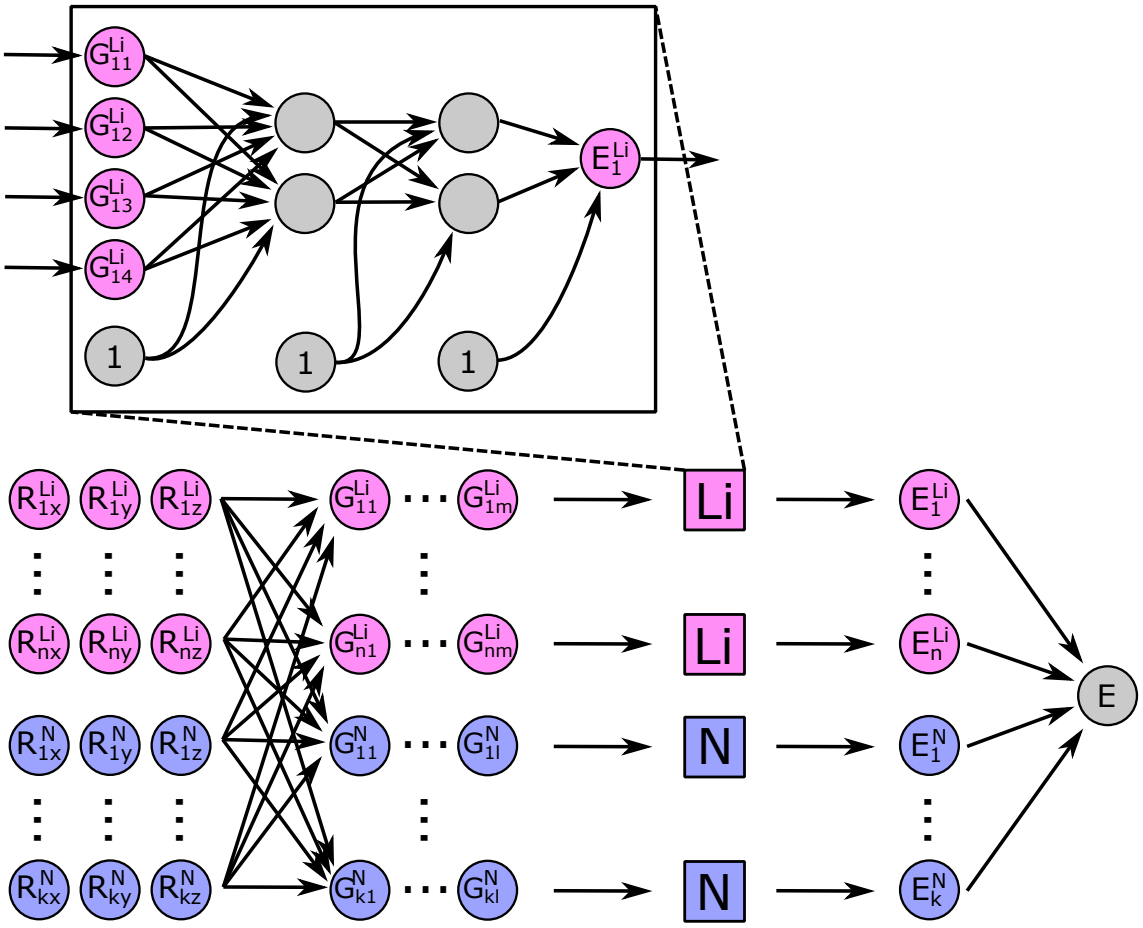


Figure 1: A schematic depiction of the High-Dimensional Neural Network Potential for a system of lithium and nitrogen.

with

$$C(x, x_l, x_c) = \begin{cases} f_p(\frac{x-x_0}{\Delta x}), & x_0 \leq x \leq x_0 + \Delta x \\ f_p(\frac{x_0-x}{\Delta x}), & x_0 - \Delta x \leq x < x_0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$C^a(x, x_l, x_c) = \begin{cases} f_p(2\left(\frac{x-x_0}{\Delta x}\right) - \left(\frac{x-x_0}{\Delta x}\right)^2), & x_0 \leq x \leq x_0 + \Delta x \\ f_p(2\left(\frac{x_0-x}{\Delta x}\right) - \left(\frac{x_0-x}{\Delta x}\right)^2), & x_0 - \Delta x \leq x < x_0 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$f_p(x) = x^3(x(15 - 6x) - 10) + 1, \tag{12}$$

where $x_0 = (x_c - x_l)/2$ and $\Delta x = (x_c + x_l)/2$. Again, these have been provided for completeness' sake, but we refer to the *n2p2* code documentation [15] for a complete overview of available symmetry functions and polynomials, along with the original publication of (weighted) polynomial symmetry functions with compact support [13].

2.2 High-Dimensional Neural Network Potential

The High-Dimensional Neural Network Potential uses several feed-forward neural networks at its core for the regression of $U_i(\mathbf{G}_i)$, one for each element in the system. A schematic

depiction can be found in Figure 1 in its application to lithium nitride. In this structure, we have n lithium and k nitrogen atoms, with m symmetry functions for lithium and l symmetry functions for nitrogen. We will now proceed to provide a brief overview of the neural network itself.

The structure of the neural network is as follows: there is an input layer, some number of hidden layers, and an output layer, each consisting of a number of nodes. The input layer necessarily has m nodes in the lithium network and l nodes in the nitrogen network; these are the values of the symmetry functions. The output layer has a single node: the *atomic* energy U (we have discarded the subscript i for the discussion of the neural network, for notational clarity). Each layer j, except the input layer, possesses an activation function $f^j(x)$, in this case usually either the hyperbolic tangent [2], $f^j(x) = \tanh x$, or the softplus function [6], $f^j(x) = \log(1 + e^x)$. Although other surjective activation functions are available, the activation function of the output layer is usually linear, $f^j(x) = x$, to be able to cover the entire $U \in (-\infty, \infty)$ range. Each node y_i^j is then related by

$$y_i^j = f^j \left(b_i^j + \sum_{k=1}^{N_{j-1}} w_{i,k}^{j-1,j} y_k^{j-1} \right), \tag{13}$$

where weights $w_{i,k}^{j-1,j}$ and biases b_i^j are free parameters and N_{j-1} is the number of nodes in the previous layer $j-1$. In the example of Figure 1,

$$U((G_1, G_2, G_3, G_4)^T) = b_1^3 + \sum_{i=1}^{2} w_{i,1}^{2,3} f^2 \left(b_i^2 + \sum_{j=1}^{2} w_{j,i}^{1,2} f^1 \left(b_j^1 + \sum_{k}^{4} w_{k,j}^{0,1} G_k \right) \right). \tag{14}$$

.

As mentioned previously, we may obtain the force on atom i by differentiating the potential energy, such that

$$\mathbf{F}_i = -\frac{\partial U}{\partial \mathbf{R}_i} \tag{15}$$

$$= -\sum_j \frac{\partial U_j}{\partial \mathbf{R}_i} \tag{16}$$

$$= -\sum_j \sum_k \frac{\partial U_j}{\partial G_{j,k}} \frac{\partial G_{j,k}}{\partial \mathbf{R}_i}, \tag{17}$$

where $\partial U_j / \partial G_{j,k}$ is obtained by the standard backpropagation method.

2.2.1 HDNNP training

Gradient descent algorithms like *Adam* [16] are the standard when training neural networks. With some slight adjustments, these can be applied here as well. In particular, we obtain a gradient

$$\frac{\partial U}{\partial w} = \sum_i \frac{\partial U_i}{\partial w}, \tag{18}$$

for some weight or bias w, where $\partial U_i / \partial w$ can be obtained through backpropagation. However, since our eventual goal is to predict forces, and forces are available from electronic

structure calculations, we can add first order information to the PES by including these in the training process. This yields the gradient

$$\frac{\partial \mathbf{F}_i}{\partial w} = \frac{\partial}{\partial w}\left(-\sum_j \sum_k \frac{\partial U_j}{\partial G_{j,k}} \frac{\partial G_{j,k}}{\partial \mathbf{R}_i}\right) \tag{19}$$

$$= -\sum_j \sum_k \frac{\partial^2 U_j}{\partial w \partial G_{j,k}} \frac{\partial G_{j,k}}{\partial \mathbf{R}_i}. \tag{20}$$

To balance the energy and force errors, we minimize the adjusted cost function

$$\Gamma = \sum_{s=1}^{N} \left(U_s - U_s^{\text{ref}}\right)^2 + \beta^2 \sum_{s=1}^{N} \sum_{i=1}^{N_s} \left|\mathbf{F}_{s,i} - \mathbf{F}_{s,i}^{\text{ref}}\right|^2, \tag{21}$$

for N structures, with structure s containing N_s atoms. We have also introduced a tunable hyperparameter β, called the force update parameter. This will be a measure of how important force errors are compared to energy errors.

It should be noted that training for the *n2p2* code, which will be used in this thesis, is not done using a gradient descent method, but rather using an Extended Kalman Filter, which introduces a number of additional training parameters (although many of these have been predetermined to work well for most systems). The filter's internal state consists of all weights and biases of the networks, with no dynamics. We refer to [6] for more details on this implementation and ideal training parameters, where its performance is also shown to be far superior for the HDNNP compared to gradient descent methods.

2.3 On-The-Fly Gaussian Overlap Potential

Another machine learning method to estimate the PES is the Gaussian Approximation Potential (GAP) [8]. This will not be covered in depth, but involves kernel ridge regression of U_i, such that

$$U_i(\mathbf{G}_i) = \sum_{i_B=1}^{N_B} w_{i_B} K(\mathbf{G}_i, \mathbf{G}_{i_B}) \tag{22}$$

for some kernel function K, a set of N_B reference descriptors $\{\mathbf{G}_{i_B}\}$, and weights $\{w_{i_B}\}$.

An advantage of this method is the fact the weight vector that minimize the error with respect to the reference structures has a closed-form solution, which means the "learning" process is much less involved than that of the HDNNP. This is leveraged in the On-The-Fly Machine-Learning Force Fields method (or simply on-the-fly method) proposed in [7], which uses a variant of GAP to completely forego the generation of an FP data set.

Training may start without any FP structures. During a training run, for every MD step, a Bayesian error estimate is used to together with a number of error conditions to determine whether new FP calculations are necessary. When this is the case, an FP calculation is performed and the MD simulation is propagated with these forces. The weights can eventually be recalculated using this new data. When the error conditions determine the GAP prediction to be good enough, however, those force predictions are used instead and FP calculations can be skipped. This way, when the training run is finished, we have produced a large MD simulation with a near-FP accuracy, but very few actual FP calculations.

2.4 On-The-Fly training sets and augmentation

In order to reduce the amount of electronic structure calculations necessary for the generation of a training set, we will use the on-the-fly method. This will generate a large data set of structures, where most structures use predicted energies and forces, but some use FP calculations. Using the latter structures to train the HDNNP completely avoids compounding the energy and force errors generated by the on-the-fly method. The only precision we require from the on-the-fly method is an identical coverage of the phase space compared to a run for which FP calculations has been used on every structure.

We may speculate those structures chosen for DFT calculations, those for which the error criteria have not been fulfilled, are in some way more important to the training process than others. This will, for example, remove structures closely related in the phase space. On the other hand, the functional form of the on-the-fly kernel ridge regression differs greatly from that of the High-Dimensional Neural Network Potential, which means some section of the PES may be easily describable with only a few data points in one representation, while the other requires many more. The training set may not contain enough information for the HDNNP to describe the PES.

When training does not lead to satisfactory results, we may be required to augment the training set with new samples. There are many ways to obtain these, but since the initial training set exhausts the available electron structure calculations, new ones will always have to be performed. Perhaps the most time-efficient method to do select new structures, is to take them from the remaining MD steps in our initial on-the-fly run, the structures for which the energy and forces were predicted by the kernel regression. With a good approximation of the time consumed by previous electronic structure calculations on the same system, the number of structures to add can be selected quite precisely depending on the time available.

Since, especially in our initial training, the energy errors produced by the HDNNP are almost definitely larger than those produced by the kernel regression, we may preferentially select structures that are currently not well represented by the HDNNP, by selecting structures for which the HDNNP energy is maximally distant from the kernel regression energy. When the HDNNP errors are of the same order of magnitude as kernel regression errors, we may switch to selecting structures for which two trained HDNNPs (with different initialization and possibly with different functional form) produce different energies [17].

For the same reason, we do not always need to set aside any of our expensive training samples for testing: a testing set can be generated from on-the-fly predictions. With sufficient memory capacity, this allows for testing sets essentially arbitrarily big compared to the training set. This, again, is only a possibility when the HDNNP errors are orders of magnitude larger than those produced by kernel regression.

We may contrast the on-the-fly training set generation to the opposite method of training size reduction, where an entire run is generated using DFT, and everything but some small random subset is thrown away [2]. This often has to be performed to reduce computation time and memory consumption. A slightly more sophisticated approach can involve a statistical analysis of redundant data [11].

2.5 Superionic conduction and lithium nitride

A superionic conductor is a solid state material exhibiting ionic conductivity on or slightly below the order of magnitude typically seen in molten salts ($\sigma \approx 1\ \Omega^{-1}\ \mathrm{cm}^{-1}$) [18]. The movement of charge carriers below the melting point in superionic conductors is typically facilitated by the diffusion of one type of ion through its sublattice, while the remaining

lattice acts as a stationary cage. These ions, then, can be said to behave more like a charged liquid than the rest of the material's solid crystalline structure.

Superionic conductors are a particular interesting class of material, thanks to the anharmonic lattice dynamics that drive the superionic phase transition [10]. Combined with the liquid-like behaviour of the ionic sublattice, this might result both in a more complicated description of the PES and a more exhaustive exploration of it, compared to a material well-described by harmonic lattice dynamics.

The material of interest to this thesis is lithium nitride (Li_3N). In this material, lithium starts to diffuse around a temperature of 550 K and experiences a large jump in diffusivity around 650 K. Throughout and above this temperature range, nitrogen acts as the stationary lattice keeping the material in solid state [10].

3 Computational methods

3.1 On-The-Fly method

All DFT calculations were performed by the *Vienna Ab initio Simulation Package* (VASP), together with the On-The-Fly Machine Learning Force Fields method as implemented by its *MLFF* feature [7]. Settings for the DFT calculations were provided by the supervisor and will not be treated in much detail. The *INCAR* files can be found in the appendix.

A diamond data set was generated using the on-the-fly method, in an NPT molecular dynamics heating run from 100 K to 3000 K in 300 picoseconds and 150000 time steps. This contains, for every structure in the MD run, the Cartesian coordinates of every atom, the force acting on every atom, and the total potential energy of the structure. The (periodic) bounding box was also included. The training set was constructed from all structures for which DFT calculations were performed, and a test was constructed from a random sample of structures for which on-the-fly calculations were performed. The *INCAR* file can be found in Appendix A.

A lithium nitride (Li_3N) data set was generated using the on-the-fly method in an NPT heating run from 150 K to 800 K in 450 picoseconds, with 300000 time steps. Again, a training set was constructed from every structure for which FP calculations were performed, and a test set was constructed from a random sample of the remaining structures. The *INCAR* file can be found in Appendix B.

A complementary lithium nitride data set was generated using the on-the-fly method in an NPT run at 750 K for 300 picoseconds and 200000 time steps. Other settings were identical the previous data set, settings can be found in Appendix C.

Subsequent DFT calculations for individual structures (as necessitated by the training set augmentation scheme) were applied with identical settings, seen in Appendix D.

3.2 Training of the HDNNP

The training of the HDNNP was performed using the *n2p2* code [6]. Analysis of results was done with assistance of the *pymatgen* [19] and *vasppy* Python libraries, as well as *pymatgen.analysis.diffusion_analyzer* [20, 21].

Diamond For this material, we have no particular interest in finding an optimal set of symmetry functions and training parameters. Standard training parameters were chosen from literature [6] and $10 + 3 \cdot 5$ type 23 (Equation 7) and type 25 (Equation 9) symmetry functions (10 radial and 15 angular, where the angular symmetry functions have 3 different radial components) were chosen to roughly minimize the energy root-mean-squared-error

(RMSE) over the testing set. The energy and force component RMSE will be the training performance metric for the remainder of this thesis. The neural network consisted of 2 hidden layers, each containing 10 nodes and a softplus activation function. The *input.nn* file can be found in Appendix E.

These symmetry functions were chosen in an equidistant grid from $r = 0$ to $r = r_c$ described by [13] (both radial symmetry functions and radial parts of angular symmetry functions), which in turn is adopted from the shifted-peak generation method in [12]. Angular symmetry functions were chosen to cover the entire 0 to π radians range, with particular attention to the presence of high-derivate functions at the tetrahedral 109.471° angle. A cut-off radius of 6 Å was chosen.

Before training, all symmetry functions whose minimum and maximum values over all atoms in the training set were no more than 10^{-3} apart, were purged using the *nnp-prune* tool in *n2p2*. This was performed for every HDNNP training procedure from this point.

During training with the *nnp-train* tool, the weights and biases of the HDNNP in each epoch are saved. When training is done, we do not necessarily select the parameters in the last epoch, since the HDNNP is prone to overfitting [2]. Rather, we select the last epoch for which both the energy and force RMSE on the test set is no larger than 1.5 times the energy and force RMSE on the training set. This method of selecting the best-performing epoch will be done after every training run from this point.

Finally, we collect every structure in the on-the-fly data set (from all MD steps, not just the training and test sets) and predict energies and forces using the *nnp-dataset* tool. We may plot the difference between these HDNNP energy and force predictions and those from the on-the-fly method, against time, as an additional error metric.

Lithium nitride A more thorough investigation into training parameters and symmetry functions was performed for this material. Several descriptors were chosen, differing in both size and radial-to-angular symmetry function ratio. These are shown in Appendix K. All possessed a cut-off radius of 6 Å, mostly determined by the size of the simulation box of the on-the-fly method. To account for the larger descriptor sizes, the neural network consisted by 20 nodes in the first layer, and 10 nodes in the second layer, again with a softplus activation function.

The $20 + 4 \cdot 10$ descriptor was chosen to perform a training run of 20 epochs for force update parameters $\beta = 5$, $\beta = 10$, and $\beta = 20$ (a range recommended by [6]). We continue with the force update parameter with the lowest energy RMSE on the test set. Training runs of 10 epochs were performed for all descriptors. Again, we continue with the descriptor with the lowest energy RMSE on the test set, and plot the energy and force errors over time relative to the on-the-fly prediction.

Using these same parameters and descriptor, we train the HDNNP on the first augmented data set (the constant-temperature lithium nitride run) and a second augmented data set. For both the same energy and force error plot was made.

The second augmented data set was generated by selecting a number of structures from the original on-the-fly heating run, those for which the energy error relative to the on-the-fly prediction is largest, and perform individual DFT calculations.

3.3 Molecular dynamics using HDNNP

Molecular dynamics for the trained HDNNPs was performed using *LAMMPS* [22] and the *n2p2* HDNNP pair-style interface.

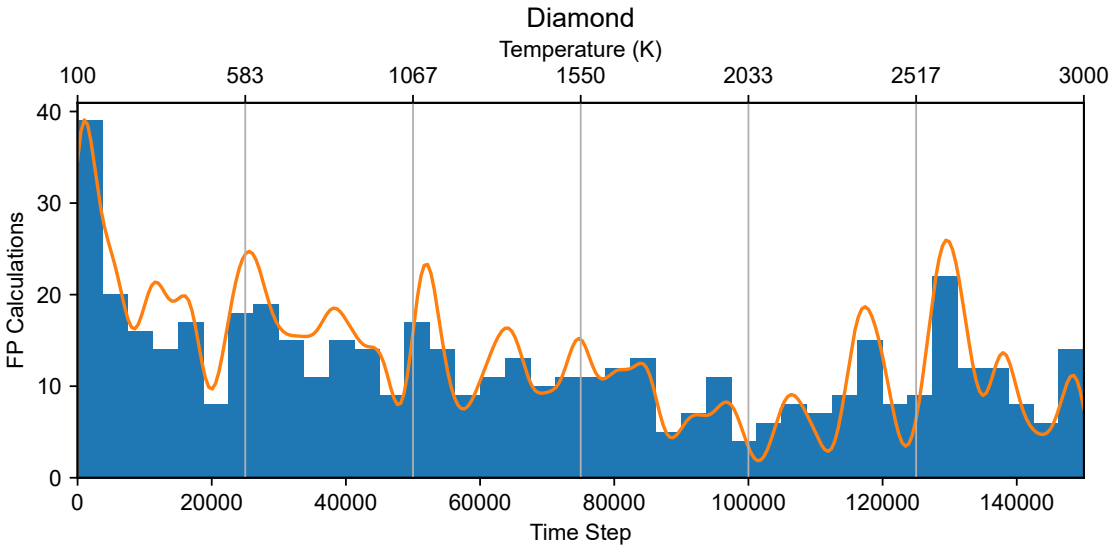


Figure 2: On-the-fly diamond heating run, density of DFT calculations. Temperature is that set by the thermostat.

Diamond An NPT heating run was performed at 0 Pa from 100 K to 3000 K in 15 picoseconds and 15000 time steps, to replicate the original on-the-fly heating run. This uses a Nose-Hoover thermo- and barostat with characteristic relaxation times of 100 femtoseconds and 1000 femtoseconds, respectively. The settings can be found in the *in.lmp* file in Appendix G.

Lithium nitride As will be discussed in further sections, not all trained HDNNPs were stable enough to perform molecular dynamics. For the original data set, we performed an NPT heating run at 0 Pa from 150 K to 400 K in 100 picoseconds and 100000 time steps, to replicate the first half of the original on-the-fly heating run. For the HDNNP trained on the first augmented data set, we performed a similar NPT heating run from 400 K to 800 K in 50 picoseconds and 50000 time steps. Again, settings in the *in.lmp* file can be found in Appendix H and I, respectively.

4 Results and discussion

We will first investigate what should be a relatively simple material: diamond. As will be shown, this material does not need any extension of the initial training set delivered by the on-the-fly method. For the verification of the HDNNP, we will compare the radial distribution functions. The second material is lithium nitride, a superionic conductor in a part of the simulated heating run. Here, we show the effects of several training set extensions.

4.1 Diamond

4.1.1 On-the-fly method

In the diamond on-the-fly heating run, out of 150000 time steps, we obtain 618 structures for which FP calculations were done to obtain energies and forces, forming the initial training set. A test set was constructed from a random sample of 182 structures (for a

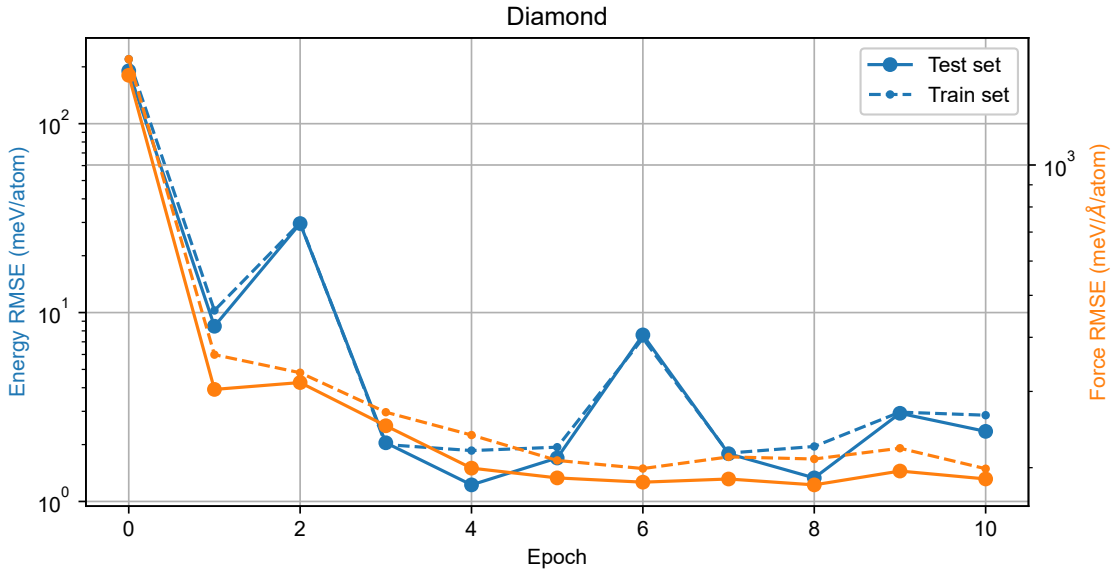


Figure 3: RMSE during training of the diamond HDNNP.

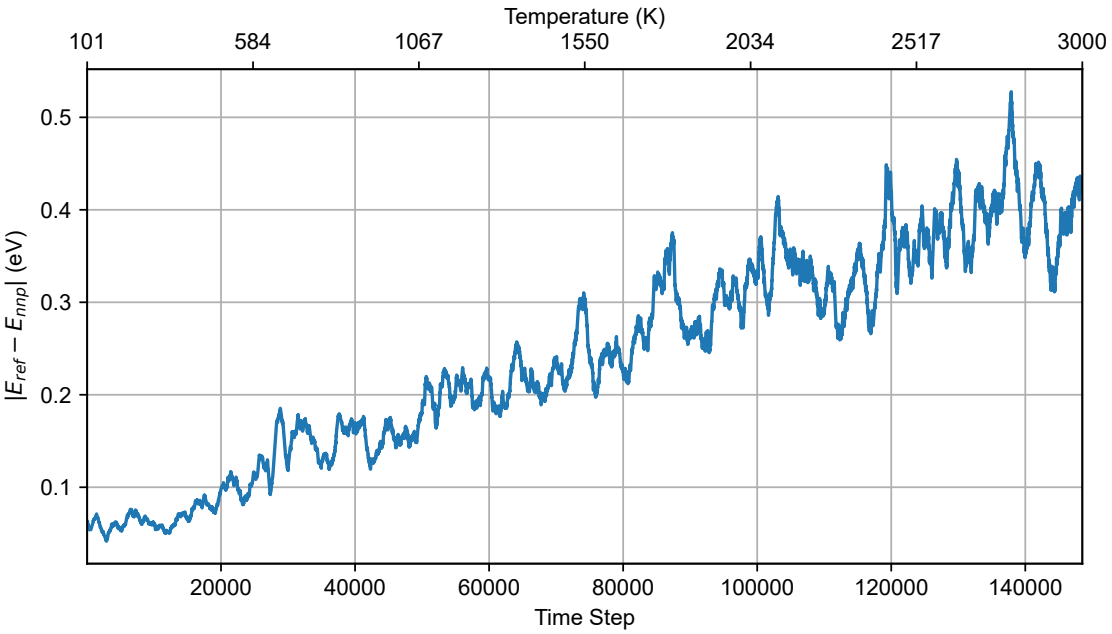
total of 800 structures, we aim for a minimum test structure count of 10% of the training structure count, but more is always better) for which on-the-fly calculations were done. We inspect the density of FP calculation in Figure 2, where it appears FP calculations are evenly spread over the heating run, apart from a major spike at the start of the simulation. Indeed, no phase transitions exist in the temperature range from 100 K to 3000 K in a vacuum, so at no point does the material explore a significantly different part of the phase space.

4.1.2 Training of HDNNP

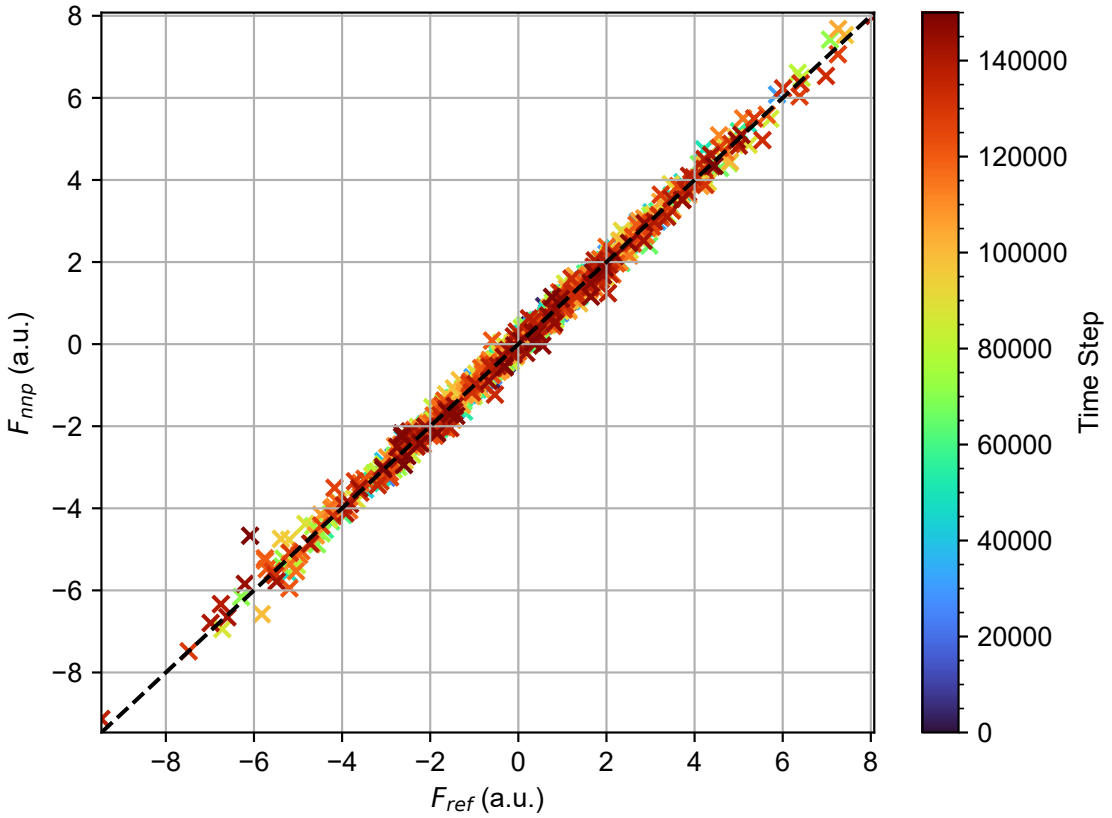
The energy and force RMSE during training can be seen in Figure 3. It is interesting to note the test error is consistently slightly below the training error. This might be caused by a number of particularly "difficult" structures, in the sense that their descriptors are similar to others but differ in atomic energy (also called contradictory data [2]). These can be explored during a transient jump in the phase space, triggering the on-the-fly method to perform FP calculations and thus excluding them from our test set. In principle, this is not bad, because we have included these difficult structures in training. However, it may be possible for the increased training set errors to hide over-fitting, which is why measures should be taken to reduce contradictory data (e.g., by providing more distinct descriptors).

The RMSE over the training and test set are not significantly different, indicating the HDNNP has not started overfitting yet. Combined with the fact the descriptor and training parameters were chosen somewhat arbitrarily, it is quite possible a much better fit is achievable. Nevertheless, this fit will be sufficient for the purpose of demonstration on this simple material, as well be showed shortly in an MD run.

Let us analyse where the error is coming from in Figure 4a, where the error in the predicted potential energy (by the trained HDNNP) is plotted, compared to that predicted by the on-the-fly method, against time. Note, this is the total potential energy of the structures, not the average atomic energy. There is a clear upwards trend as temperature rises. This is most likely the result of the system exploring an increasingly large, and increasingly sparsely sampled, phase space as vibration become more pronounced. A sim-



(a) Total potential energy difference between HDNNP and on-the-fly prediction. Temperature is that set by the thermostat.



(b) A random sample of HDNNP force component predictions against on-the-fly predictions.

Figure 4: Trained HDNNP for diamond. Energy and force component errors over entire on-the-fly MD run, with respect to on-the-fly predictions.

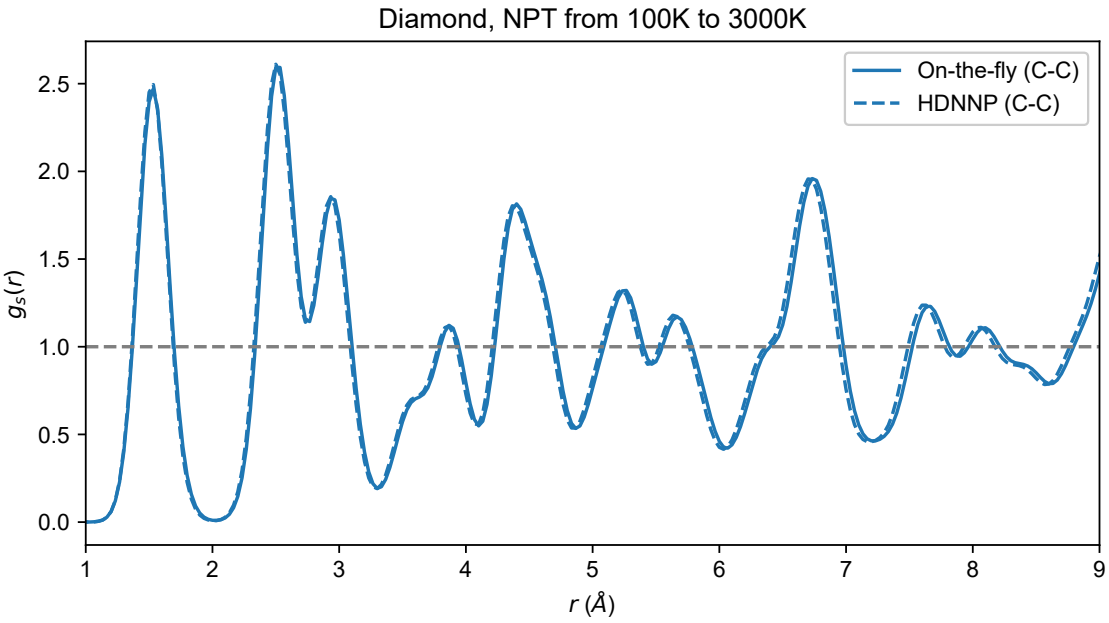


Figure 5: Diamond radial distribution function as predicted by on-the-fly method and as predicted by the trained HDNNP.

ilar plot is made in Figure 4b with a random sample of forces instead of total potential energies. Since the forces are relatively accurate (that is, most forces lie on the diagonal $F_{nnp} = F_{ref}$), the trend is less obvious, but forces particularly far from the diagonal are mostly those in later time steps.

4.1.3 Verification

Finally, to verify our result for diamond, we perform a molecular dynamics run with the trained HDNNP and compare the radial distribution function of carbon to the on-the-fly method in Figure 5. The two agree almost perfectly, and the HDNNP seems to have fully captured the dynamics of carbon atoms using only the 618 structures provided by the on-the-fly method.

4.2 Lithium nitride

4.2.1 On-the-fly method

The lithium nitride on-the-fly heating run, ran for 300000 time steps. Of those, FP calculations were performed for 461 structures, forming the initial training set. Again, we observe where the electronic structure calculations are done for this heating run in Figure 6. The phase transition is immediately apparent: around time step 200000 (where the thermostat is set to about 600 K), there is a massive increase in calculations. Later at 250000 (where the thermostat is set to about 700 K), we see a similar spike. Lithium diffusion starts to occur above a temperature of 500 K, and is greatly increased above 600 K [10].

Another on-the-fly lithium nitride run was done at a temperature of 750 K. This was done on top of the previous run and yielded an additional 193 structures for which FP calculations were performed. These will be added later to form the first augmented data set.

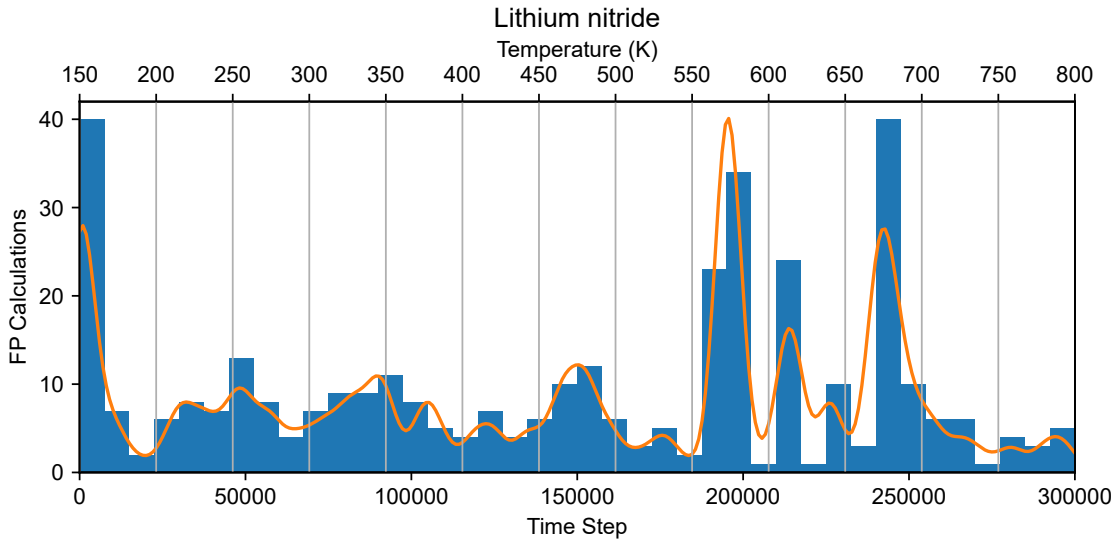


Figure 6: on-the-fly lithium nitride heating run, density of DFT calculations. Temperature is that set by the thermostat.

4.2.2 Training of HDNNP

Initial training set A test set was formed from 139 structures randomly selected from the remaining on-the-fly calculations. This brings to total to 600 structures, 461 for training and 139 for testing.

As discussed, one of the most important parameter for training is the force update parameter. Using the advised range of values in [6], several runs using the $20 + 4 \cdot 10$ descriptor relieved $\beta = 20$ to yield satisfactory energy RMSE while keeping over-fitting to a minimum. The descriptor was further analysed by using the obtained parameters with 7 different descriptors. Test errors per epoch can be seen in Figure 7.

It is obvious the descriptor does not have great effect on the training process at these sizes. This indicates a more systematic problem in predicting the PES than the descriptor's ability to uniquely represent all available structures. It might originate from the relatively small training set size of 461 structures, which can be uniquely represented by a few well-chosen symmetry functions. As a result, choosing the absolute smallest descriptor we can afford might not be the best option if we intend to augment the training set size later. Even though choosing a larger-than-necessary descriptor may be detrimental to performance, it can avoid contradictory data in new structures. Alternatively, we may expand the descriptor upon introducing new data [2], though we have not done that here. Combining the concept of a gradually expanding descriptor with a more sophisticated and automated method of finding descriptors would be an interesting prospect to automate the entire training and data set augmentation cycle.

We find a good compromise between descriptor size and performance at 60 symmetry functions per descriptor. A radial-to-angular symmetry function ratio of $1 : 1$ has previously been found to work well across data sets [12] (indeed descriptors with many angular symmetry functions seem to perform slightly worse in Figure 7), so we will be continuing with the $30 + 3 \cdot 10$ descriptor and taking parameters from epoch 9, given that epoch 10 seems to have slightly over-fitted.

Before expanding the data set, let us do the same analysis of energy error as done for diamond. In Figure 8a, the HDNNP energy error with respect to the on-the-fly prediction

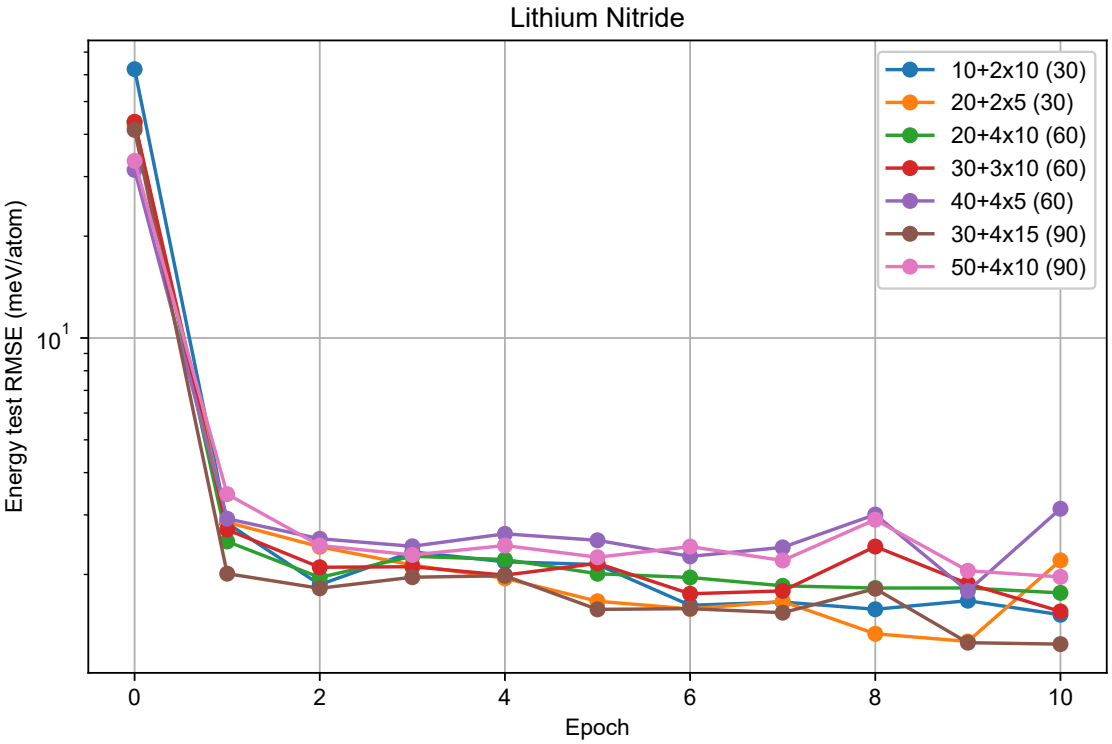
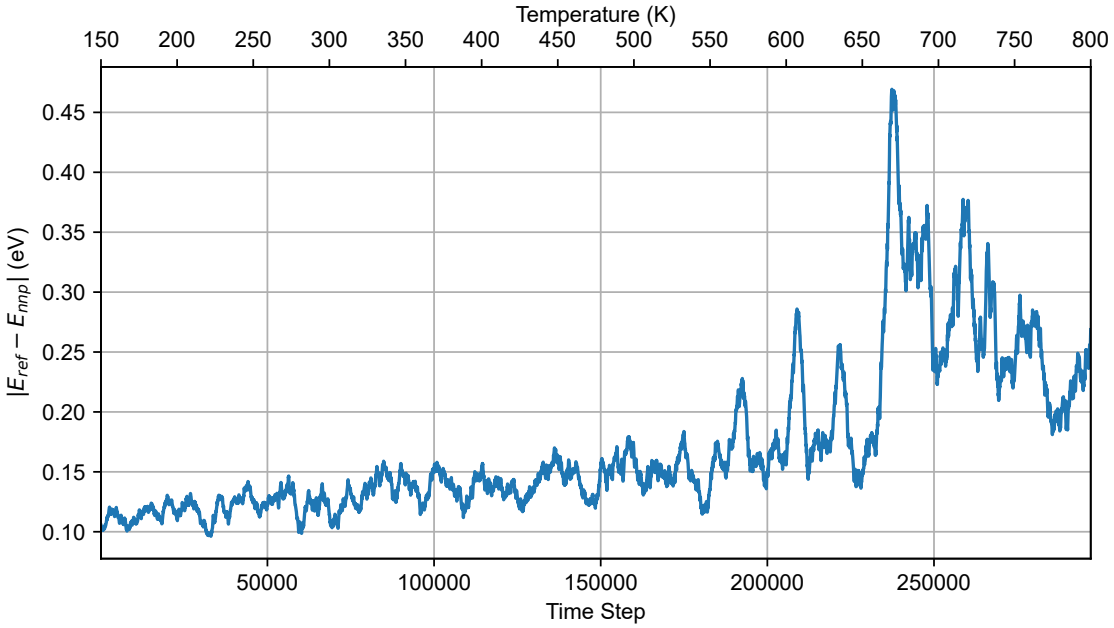
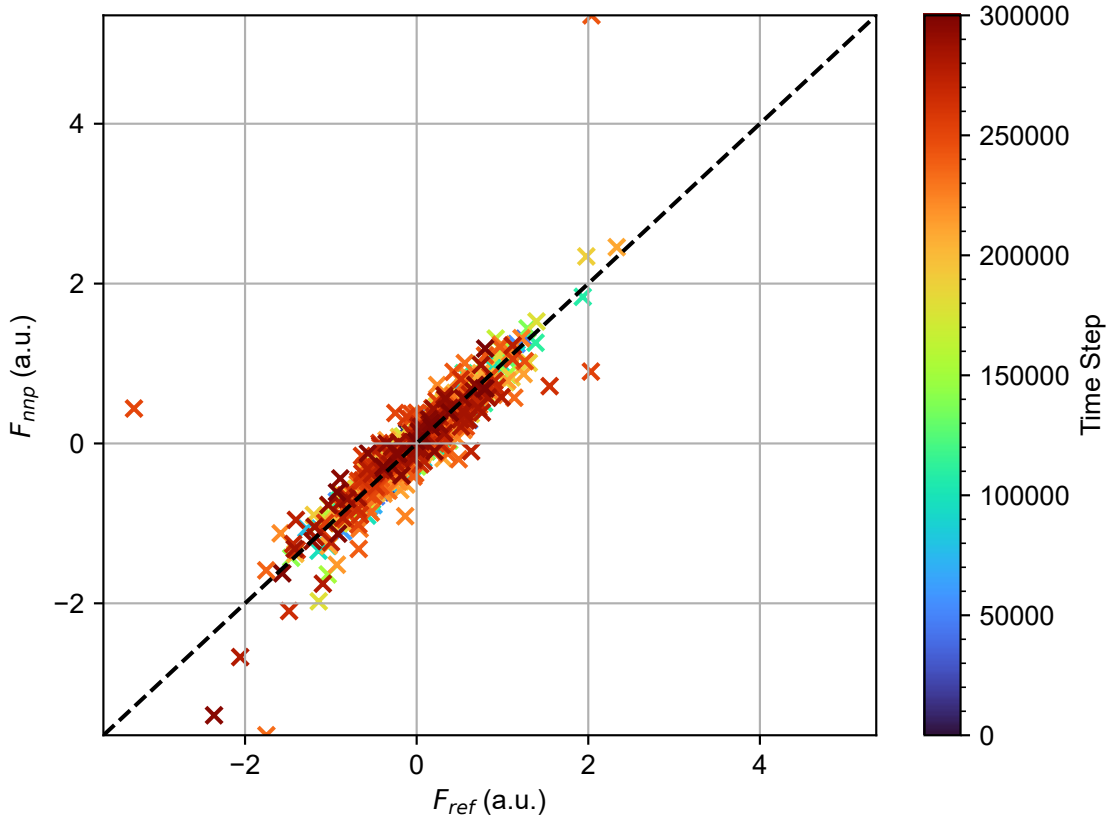


Figure 7: Energy RMSE on test set during training of lithium nitride HDNNP, for seven different descriptors. Denoted $N_r + N_{ar} \cdot N_a$, with N_r radial symmetry functions, and $N_{ar} \cdot N_a$ angular symmetry functions with N_{ar} radial parts and N_a angular parts. Exact parameters can be found in Appendix K.

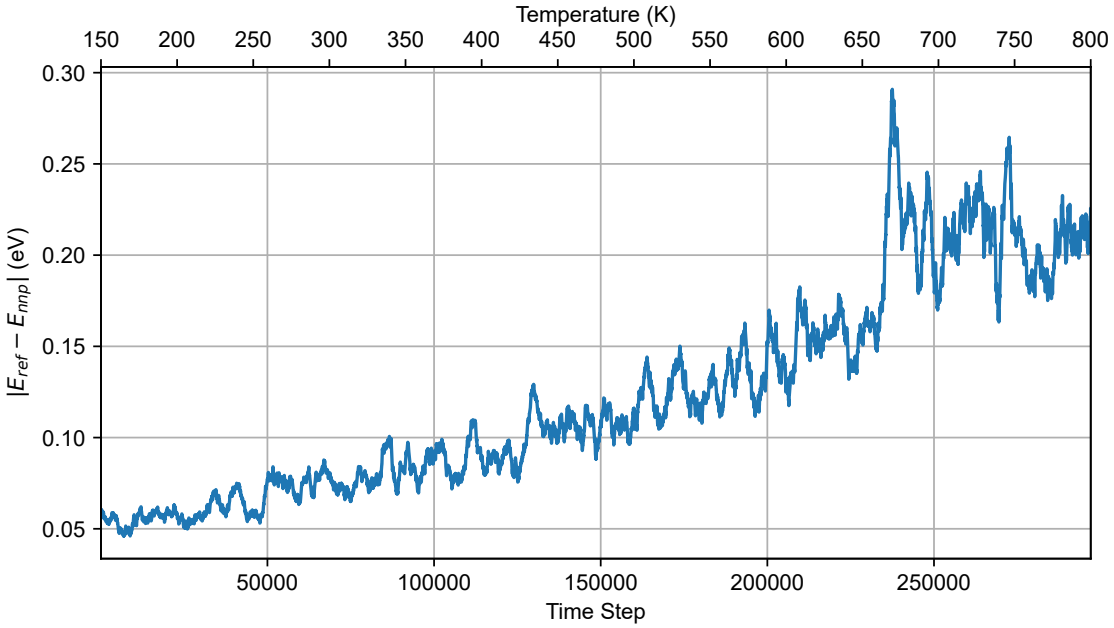


(a) Total potential energy difference between HDNNP and on-the-fly prediction. Temperature is that set by the thermostat.

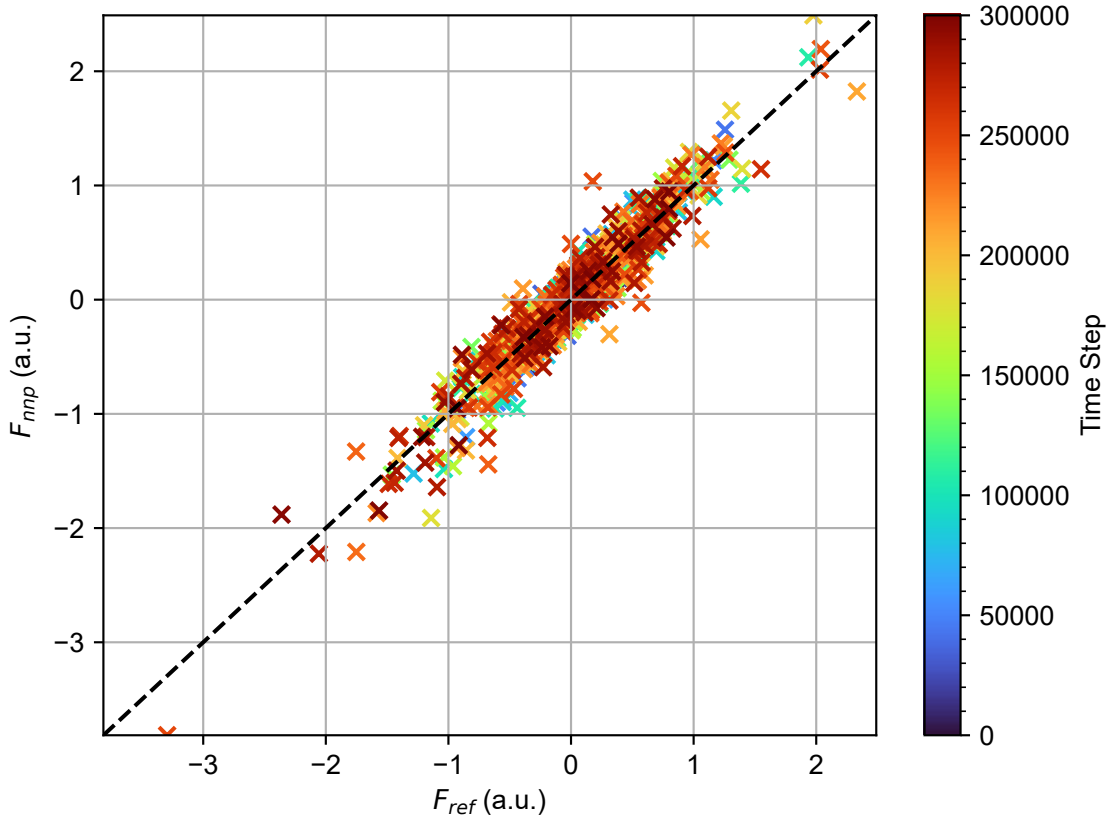


(b) A random sample of HDNNP force component predictions against on-the-fly predictions.

Figure 8: Trained HDNNP for lithium nitride, using original on-the-fly training set. Energy and force component errors over entire on-the-fly MD run, with respect to on-the-fly predictions.

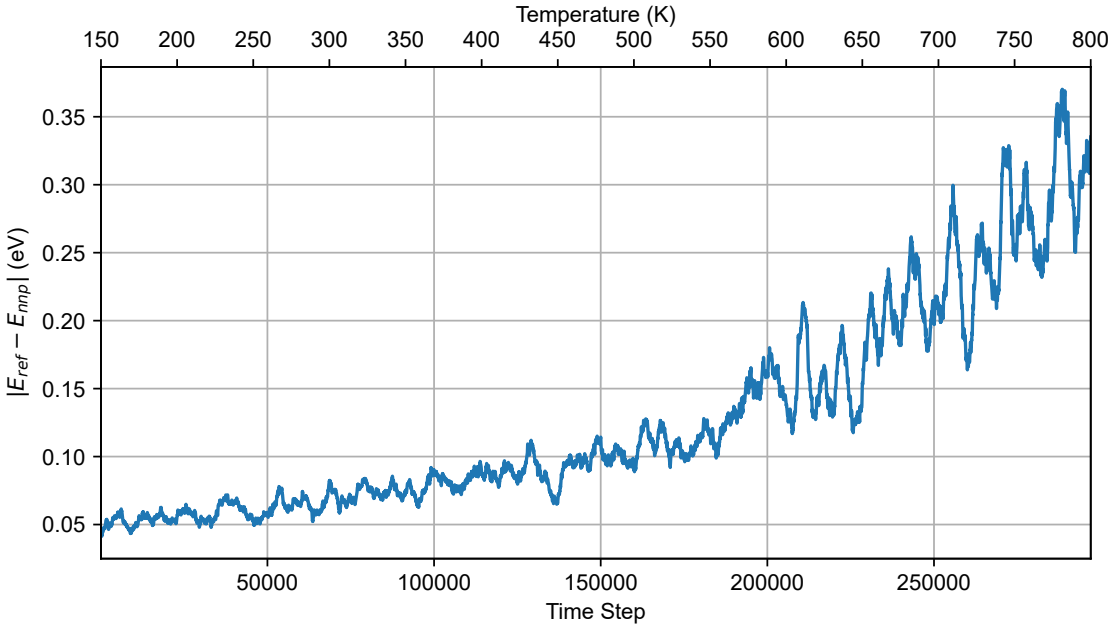


(a) Total potential energy difference between HDNNP and on-the-fly prediction. Temperature is that set by the thermostat.

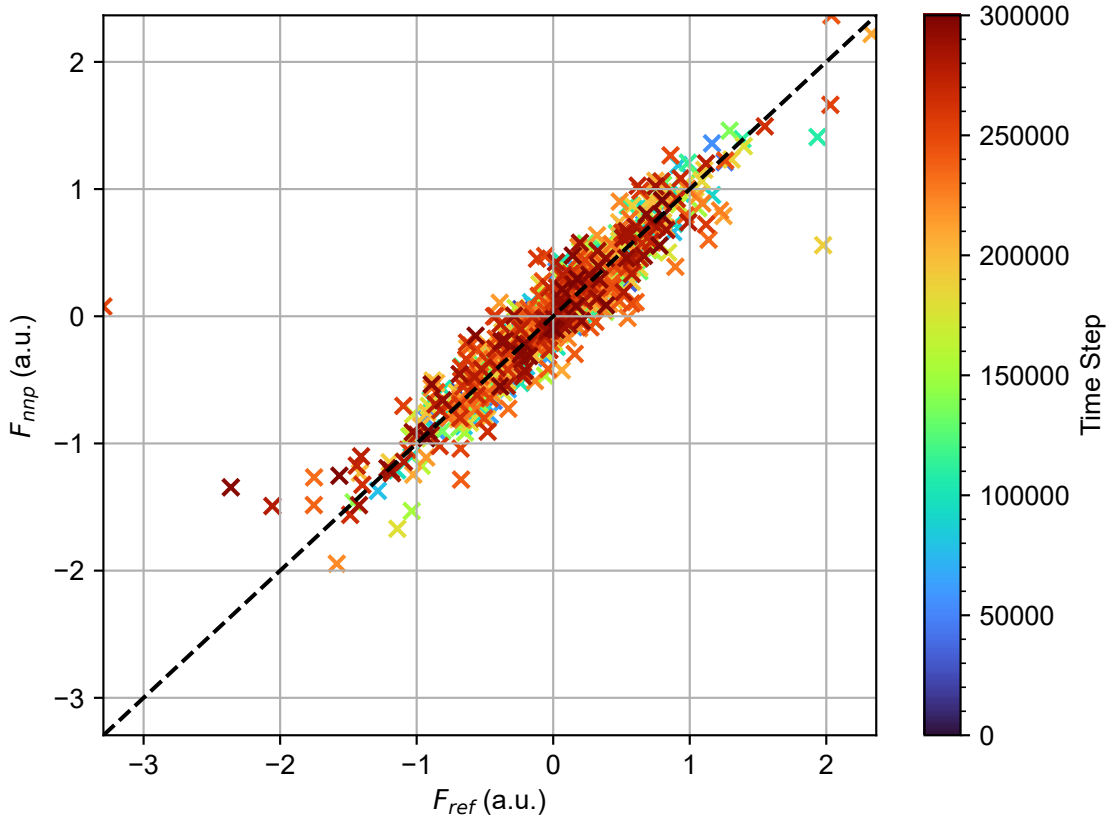


(b) A random sample of HDNNP force component predictions against on-the-fly predictions.

Figure 9: Trained HDNNP for lithium nitride, using original on-the-fly training set augmented by on-the-fly 750 K training set. Energy and force component errors over entire on-the-fly MD run, with respect to on-the-fly predictions.



(a) Total potential energy difference between HDNNP and on-the-fly prediction. Temperature is that set by the thermostat.



(b) A random sample of HDNNP force component predictions against on-the-fly predictions.

Figure 10: Trained HDNNP for lithium nitride, using original on-the-fly training set augmented by high-energy-error structures. Energy and force component errors over entire on-the-fly MD run, with respect to on-the-fly predictions.

is plotted against time. While the region under the phase transition exhibits the same behaviour as diamond, with a linear increase in error as temperature rises, a substantial increase in error is seen around and above the phase transition. The same can be seen in Figure 8b for the forces, where outliers are even more apparent due to the lack of smoothing. This suggests it is specifically the dynamics of lithium diffusion which the HDNNP is unable to capture using the initial data set.

First augmented training set As previously mentioned, there are several options available for expanding the training set. Since the initial training set shows relatively large errors above the phase transition, we will first consider adding training structures from another on-the-fly run done in this troubling region. For this, a temperature of 750 K was chosen, a superionic conducting-temperature still well below the melting point of about 814 K. Note, as discussed in Section 4.2.1, this resulted in 193 additional structures. A test set of 155 structures was randomly selected from the remaining on-the-fly predictions, making for 800 structures, 645 for training and 155 for testing. We train with the exact same parameters as the original training set, for ten epochs. The trained HDNNP was then tested on the entire on-the-fly data set again, as shown in Figure 10.

Below the phase transition, energy errors do not seem much improved, although we now rise from 0.05 eV to 0.15 eV, instead of the 0.10 eV to 0.15 eV observed previously. Indeed, we do not expect this region to see much improvement; the additional high-temperature structures were disturbed from their low-temperature states, such that they cover different portions of the phase space. The improvement in error for the very-low-temperature states may for the most part originate from run-to-run variances caused by initialization, although, of course, a better fit for high-temperature structures is not completely uncorrelated from a better fit for low-temperature structures.

More notably, the peak in energy error as the structures enter the superionic phase transition is still present, but much less notable. And the far outliers in the force component errors previously seen in Figure 8b have been removed altogether. It remains to be analysed whether this reduction is sufficient to properly simulate the material in its superionic conducting phase.

Second augmented training set We will now use the maxima in energy error as shown in Figure 8a to select new structures to be added to the training set. We have decided to double our original training set with 461 new structures. We also forced to reduce the number of test structures to 98, to minimize performance and memory impact. This makes for a total of 1020 structures, 922 for training and 98 for testing.

We see the resulting error on the entire heating run in Figure 10. The error does rise above levels seen in Figure 9, but now exhibits the same behaviour as would be expected from a simple rise in temperature; the energy error rises linearly. Forces have also regained some outliers. The change in slope at the phase transition temperature might be due to new dynamics becoming available to the lithium atoms, besides the harmonic vibration modes that is has in its solid crystalline structure. Perhaps the most significant aspect is the fact the HDNNP does not experience a jump in energy error as it did before. This means, at the very least, the energies during low-temperature lithium diffusion are being estimated with an accuracy not much less than that below the phase transition.

4.2.3 Verification

Initial training set To verify the performance of the HDNNP, a variety of MD runs were done with each trained potential. Since the energy error has remained quite similar

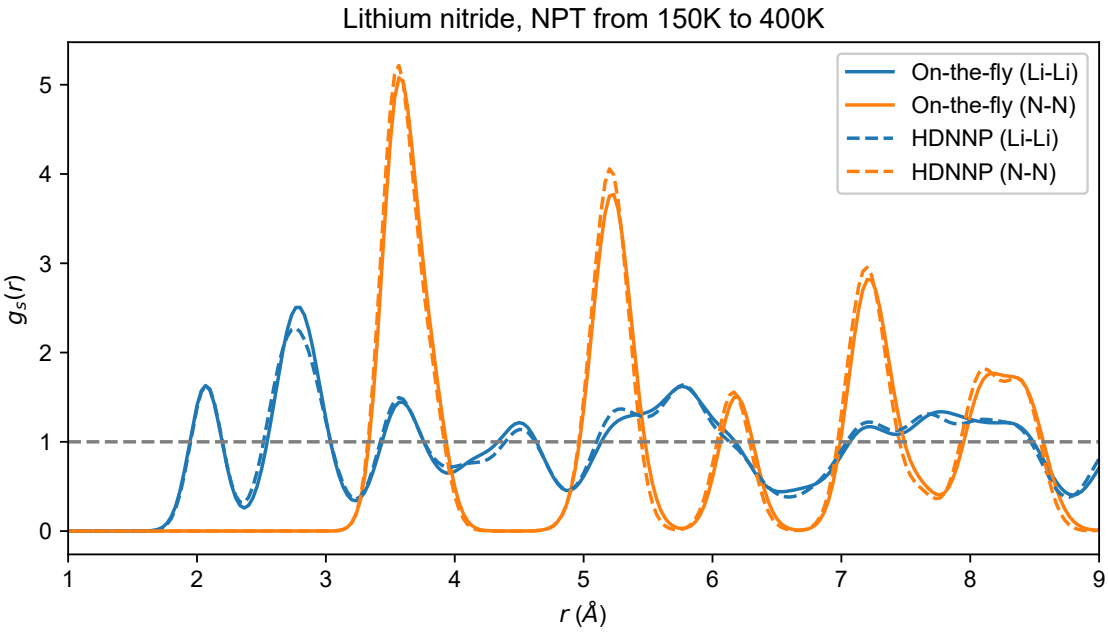


Figure 11: Lithium nitride radial distribution function for the 150 K to 400 K temperature range, as predicted by on-the-fly method and as predicted by the trained HDNNP.

between the different training sets below the phase transition, we might be interested whether this region is already correctly modelled in the initial training set. Figure 11 shows the radial distribution function between a temperature of 150 K and 400 K, as predicted by the HDNNP in an MD run.

Performing MD runs above the phase transition is difficult, because the system quickly becomes unstable, with most runs breaking in the 400 K to 600 K range during a heating run. In most cases, a lithium atoms can be seen drifting too close to others before shooting off and disturbing the rest of the system. Although we have correctly captured the existence of a phase transition, the HDNNP does not appear to be close to capturing its temperature, let alone the diffusion behaviour above the phase transition.

First augmented training set This dataset performed much better in the 400 K to 800 K temperature range. The system is still very unstable, the molecular dynamics run from 400 K to 800 K stopped around 640 K, switch seems to be around the usual temperature before breaking. The mean squared displacement has been plotted in Figure 12. There is definitely a semblance of diffusive behaviour over time, but it is intermittent, at some points linearly rising as would be expected from diffusion, and at some points jumping up, indicating a transient restructuring of the system. The mean squared displacement of nitrogen is barely increasing, as expected. It has also correctly simulated a stronger in-plane lithium diffusion (x- and y-directions) than out-of-plane diffusion (z-direction) [10].

There are a number of possible causes for the jumping behaviour. The heating run might be too short, meaning the restructuring of the system is caused by the thermostat kicking an atom out of its normal range for that temperature. This might be kept in check in future runs with a longer characteristic relaxation time, or by simply increasing the relatively small number of time steps (currently 30000). Of course, it is also likely the diffusion is simply not captured very well by the HDNNP.

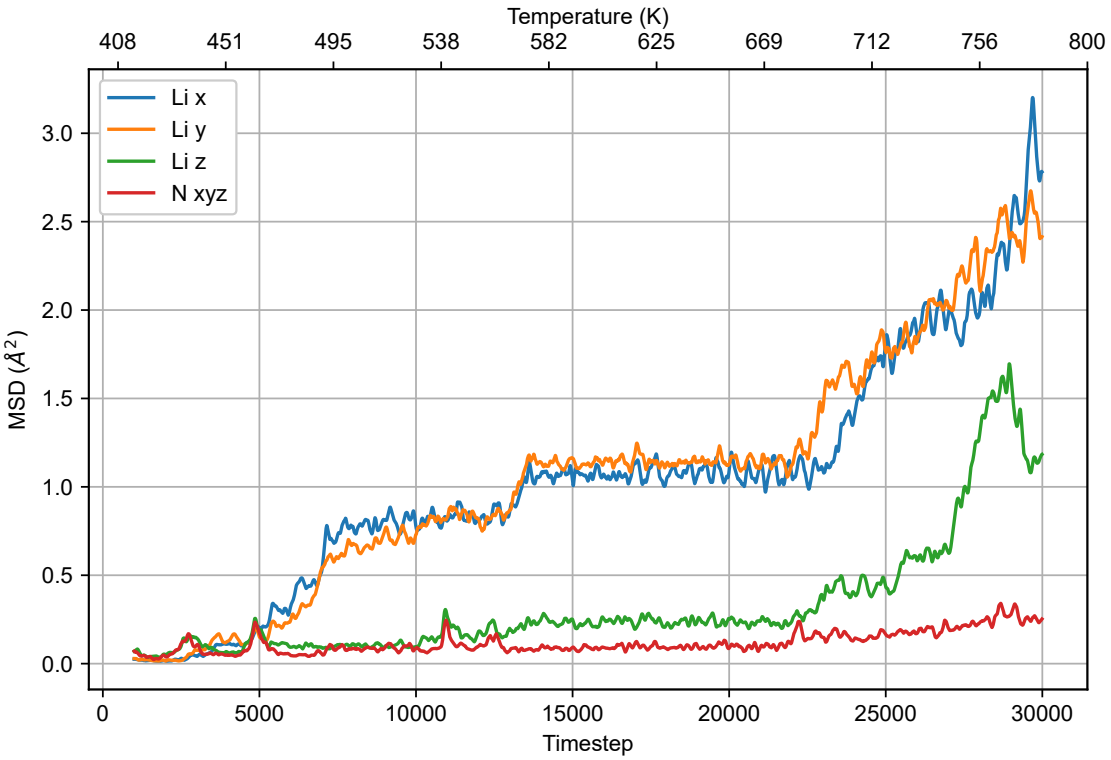


Figure 12: Lithium nitride mean-squared-displacement over NPT heating run using trained HDNNP. Temperature is that set by the thermostat.

Second augmented training set Although the energy error graph seemed promising, high-temperature runs with this training set again become too unstable in the superionic phase to run molecular dynamics for any reasonable amount of time steps. This might be related to the force outliers seen returning in Figure 10b.

5 Conclusions

The method presented in this report seems promising. For the simple case of diamond, the HDNNP had no problem achieving a good fit on the PES, even with what is almost certainly a suboptimal selection of descriptor, network topology, and training parameters. This is the case throughout the simulated temperature range of 100 K to 3000 K. Because of the on-the-fly training set generation, electron structure calculations were only performed 461 times, out of 150000 time steps, which is a 325-fold reduction in computation time.

Lithium nitride, however, is much more difficult to model with an HDNNP in its superionic phase, using the on-the-fly training set. The initial training set produces reasonable predictions below the superionic phase transition, but the HDNNP is complete unusable above it. Augmenting the training set to about one-and-a-half times its original size shows a promising reduction in energy errors, and begins to qualitatively approximate expected diffusive behaviour in the material during molecular dynamics. Going beyond this, to about twice the original training set size, the energy error ceases to show significant jumps as the material enters the superionic phase transition. However, inaccurate force predictions again make the HDNNP too unstable to perform molecular dynamics.

We were unable to train the HDNNP to simulate lithium nitride and produce diffusivity

measurements that can be compared against empirical measurements. But the proposed technique shows promising limiting behaviour and, with proper attention to descriptor selection and training parameters, seems to be exceedingly close to correctly capturing superionic conduction.

6 Outlook

There are a number of improvements to the analysis done in this thesis that would significantly improve the applicability of the results. We will finish by listing some areas requiring further investigation.

First and foremost, due to time constraints, further analysis of data set expansion was not possible. The largest data set, the second augmented data set, was also running against the hardware limitations of the machine running these training runs. Memory consumption is large, and although there have been many efforts to mitigate this [6], going beyond these sizes simply requires the use of High Performance Computing. This means the precise amount of additional structures required to perform MD runs of the quality as the on-the-fly method remains elusive. And while this work suggests the existence of such a limit, future work is required to achieve it.

Comparing the diamond energy error in Figure 4a with any of the three lithium nitride energy errors in Figure 8a, Figure 9a, or Figure 10a, it is interesting to note energy errors are nearly identical, and often lower in the lithium nitride HDNNPs. This despite lithium nitride having significantly worse force predictions. This might indicate energy errors have been prioritized too much during training, which can be caused either by the force update parameter β being too low, or the force-to-energy-update fraction in the training process being too low, in turn not providing the training process with enough force updates to achieve satisfactory errors, even with a correctly tuned force update parameter. A more in depth investigation into training parameters is warranted for lithium nitride.

There is also a possible problem arising from the training set augmentation scheme, where the HDNNP-to-on-the-fly error is used to select new structures. Note Figure 8a is smoothed with a moving average, which reveals the trend in error. While the general trend in the figure can be used to identify regions where the descriptors and training parameters are systematically not sufficient and do not hold enough information to capture the dynamics of the structures in that region, the specific indices of maximum peaks in the figure are most likely highly dependent on the initialization of the HDNNP, and correspond only to structures that are coincidentally badly fitted. In the case of lithium nitride, for example, it is not particularly worrying when such a peak lies in the region above the phase transition, but it could cause peaks below the transition, which do not actually appear to be of much interest. Mitigation methods worth investigation could involve first smoothing the energy error, or averaging the error over multiple training runs with differing initializations. The latter, particularly, could result in much more consistent energy error graphs.

Another notable problem is the fact the augmented training sets were generated both in different amounts and with different methods; one systematically selected new structures from the original MD run, and the other starting new MD runs in an ad hoc manner. Especially the latter ad hoc method may cause the user to introduce a bias in the data set towards temperature ranges of interest to them, decreasing the HDNNP's ability to train on other regions. It would be interesting to see both of these methods expanded to larger training set sizes, especially with regard to the performance across temperature ranges.

References

[1] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98, 4 2007.

[2] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. volume 115, pages 1032–1050. John Wiley and Sons Inc, 8 2015.

[3] Jörg Behler. Perspective: Machine learning potentials for atomistic simulations. *Journal of Chemical Physics*, 145, 11 2016.

[4] Jörg Behler. Four generations of high-dimensional neural network potentials. *Chemical Reviews*, 121:10037–10072, 8 2021.

[5] Emir Kocer, Tsz Wai Ko, and Jörg Behler. Neural network potentials: A concise overview of methods. 7 2021.

[6] Andreas Singraber, Tobias Morawietz, Jörg Behler, and Christoph Dellago. Parallel multistream training of high-dimensional neural network potentials. *Journal of Chemical Theory and Computation*, 15:3075–3092, 5 2019.

[7] Ryosuke Jinnouchi, Jonathan Lahnsteiner, Ferenc Karsai, Georg Kresse, and Menno Bokdam. Phase transitions of hybrid perovskites simulated by machine-learning force fields trained on the fly with bayesian inference. *Physical Review Letters*, 122, 6 2019.

[8] Albert P. Bartók and Gábor Csányi. Gaussian approximation potentials: a brief tutorial introduction. 2 2015.

[9] Bartomeu Monserrat, N. D. Drummond, and R. J. Needs. Anharmonic vibrational properties in periodic systems: Energy, electron-phonon coupling, and stress. *Physical Review B*, 87(14), 2013.

[10] Gabriel Krenzer, Chang-Eun Kim, Kasper Tolborg, Benjamin J. Morgan, and Aron Walsh. Anharmonic lattice dynamics of superionic lithium nitride. *Journal of Materials Chemistry A*, 10(5):2295–2304, Oct 2022.

[11] Giulio Imbalzano, Andrea Anelli, Daniele Giofré, Sinja Klees, Jörg Behler, and Michele Ceriotti. Automatic selection of atomic fingerprints and reference configurations for machine-learning potentials. *Journal of Chemical Physics*, 148, 6 2018.

[12] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand. Wacsf - weighted atom-centered symmetry functions as descriptors in machine learning potentials. *Journal of Chemical Physics*, 148, 6 2018.

[13] Martin P. Bircher, Andreas Singraber, and Christoph Dellago. Improved description of atomic environments using low-cost polynomial functions with compact support. 10 2020.

[14] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *Journal of Chemical Physics*, 134, 2 2011.

[15] N2p2 documentation. `https://compphysvienna.github.io/n2p2/`. Accessed: 2022-07-02.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2022.

[17] Jörg Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *Journal of Physics Condensed Matter*, 26, 5 2014.

[18] J B Boyce and B A Huberman. Superionic conductors: Transitions, structures, dynamics.

[19] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L. Chevrier, Kristin A. Persson, Gerbrand Ceder, and et al. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[20] Yifei Mo, Shyue Ping Ong, and Gerbrand Ceder. First principles study of the li10gep2s12 lithium super ionic conductor material. *Chemistry of Materials*, 24(1):15–17, 2011.

[21] Shyue Ping Ong, Yifei Mo, William Davidson Richards, Lincoln Miara, Hyo Sug Lee, and Gerbrand Ceder. Phase stability, electrochemical stability and ionic conductivity of the li10±1mp2x12(m = ge, si, sn, al or p, and x = o, s or se) family of superionic conductors. *Energy Environ. Sci.*, 6(1):148–156, 2013.

[22] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

A Diamond INCAR

```
#####################################
### Parameters for VASP       ###
#####################################
SYSTEM = Diamond
SIGMA = 0.01 ; ISMEAR = 0
ENCUT =  400
NCORE = 4 ; KPAR = 8
EDIFF = 1E-4 ; NELMIN = 6
ISYM = 0
LREAL = A
IBRION = 0
ISIF = 3
MDALGO=3                  # use Langevin thermostat
LANGEVIN_GAMMA = 1.0  # friction coef. for atomic DoFs for each species
LANGEVIN_GAMMA_L=3.0   # friction coef. for the lattice DoFs
PMASS=100                 # mass for lattice DoFs
LATTICE_CONSTRAINTS = T T T   # fix x&y, release z lattice dynamics
PSTRESS=0.001             # P is set at 0.001 KB.
IBRION = 0
SMASS = 0
POTIM = 2.00
TEBEG = 100.00D0
TEEND = 3000.00D0
NSW = 150000
NBLOCK = 1
NWRITE = 1
NELM = 100
ISPIN = 1
INIWAV = 1
IWAVPR = 1
ISTART = 0
LWAVE = .FALSE.
LCHARG = .FALSE.
#####################################
### MACHINE-LEARNING          ###
#####################################
### General parameters        ###
ML_LMLFF = .TRUE.  # Set as LMLFF_FF = .TRUE., when some machine-learning force field
    calculations are executed. Default is .FALSE.
ML_ISTART = 0       # Parameter to control restarting calculations. Set 0, 1 or 2.
                    # When you execute the training from the scratch, this needs to be set to
                        0.
                    # When you restart the training reading the previous ab initio data (ABCAR
                        file), this needs to be set to 1.
ML_RCUT1 = 6.0          # Cutoff radius used for calculating the radial descriptor. Default is
    RCUT2_FFM.
ML_RCUT2 = 5.0          # Cutoff radius used for calculating the angular descriptor. Default is 5.0
     (Angst).
#ML_NHYP1 = 1           # Hyper parameter used for calculating the radial descriptor. Default is
    1.
#ML_NHYP2 = 4           # Hyper parameter used for calculating the angular descriptor. Default is
    4.
ML_MB=2000
ML_MCONF= 1600
```

B Lithium nitride (heating) INCAR

```
SYSTEM =Li3N
PREC = FAST
ALGO = Fast
SIGMA = 0.01 ; ISMEAR = 0
ENCUT =  350
NCORE = 4 ; KPAR =8
EDIFF = 1E-4 ; NELMIN = 6
ISYM = 0
LREAL = A
ISIF = 3
MDALGO=3                 # use Langevin thermostat
LANGEVIN_GAMMA = 1.0 1.0   # friction coef. for atomic DoFs for each species
LANGEVIN_GAMMA_L=3.0   # friction coef. for the lattice DoFs
PMASS=100                # mass for lattice DoFs
LATTICE_CONSTRAINTS = T T T   # fix x&y, release z lattice dynamics
PSTRESS=0.001            # P is set at 0.001 KB.
POTIM = 1.5 #2.50
IBRION = 0
SMASS = 0
TEBEG = 150.00D0
TEEND = 800.00D0
NSW = 300000 #xx K/ps
NBLOCK = 1
NWRITE = 1
NELM = 100
ISPIN = 1
INIWAV = 1
IWAVPR = 1
ISTART = 0
LWAVE = .FALSE.
LCHARG = .FALSE.
#################################
### MACHINE-LEARNING          ###
#################################
### General parameters        ###
ML_LMLFF = .TRUE.  # Set as LMLFF_FF = .TRUE., when some machine-learning force field
    calculations are executed. Default is .FALSE.
ML_ISTART = 0       # Parameter to control restarting calculations. Set 0, 1 or 2.
                    # When you execute the training from the scratch, this needs to be set to
                        0.
                    # When you restart the training reading the previous ab initio data (ABCAR
                        file), this needs to be set to 1.
ML_RCUT1 = 6.0          # Cutoff radius used for calculating the radial descriptor. Default is
    RCUT2_FFM.
ML_RCUT2 = 5.0          # Cutoff radius used for calculating the angular descriptor. Default is 5.0
     (Angst).
#ML_NHYP1 = 1           # Hyper parameter used for calculating the radial descriptor. Default is
    1.
#ML_NHYP2 = 4           # Hyper parameter used for calculating the angular descriptor. Default is
    4.
ML_MB=2000
ML_MCONF= 1600
```

C Lithium nitride (750K) INCAR

```
SYSTEM =Li3N
PREC = FAST
ALGO = Fast
SIGMA = 0.01 ; ISMEAR = 0
ENCUT =  350
NCORE = 4 ; KPAR =8
EDIFF = 1E-4 ; NELMIN = 6
ISYM = 0
LREAL = A
ISIF = 3
MDALGO=3                # use Langevin thermostat
LANGEVIN_GAMMA = 1.0 1.0   # friction coef. for atomic DoFs for each species
LANGEVIN_GAMMA_L=3.0   # friction coef. for the lattice DoFs
PMASS=100               # mass for lattice DoFs
LATTICE_CONSTRAINTS = T T T   # fix x&y, release z lattice dynamics
PSTRESS=0.001           # P is set at 0.001 KB.
POTIM = 1.5 #2.50
IBRION = 0
SMASS = 0
TEBEG = 750.00D0
TEEND = 750.00D0
NSW = 200000 #xx K/ps
NBLOCK = 1
NWRITE = 1
NELM = 100
ISPIN = 1
INIWAV = 1
IWAVPR = 1
ISTART = 0
LWAVE = .FALSE.
LCHARG = .FALSE.
#####################################
### MACHINE-LEARNING          ###
#####################################
### General parameters        ###
ML_LMLFF = .TRUE.  # Set as LMLFF_FF = .TRUE., when some machine-learning force field
    calculations are executed. Default is .FALSE.
ML_ISTART = 1       # Parameter to control restarting calculations. Set 0, 1 or 2.
                    # When you execute the training from the scratch, this needs to be set to
                        0.
                    # When you restart the training reading the previous ab initio data (ABCAR
                        file), this needs to be set to 1.
ML_RCUT1 = 6.0          # Cutoff radius used for calculating the radial descriptor. Default is
    RCUT2_FFM.
ML_RCUT2 = 5.0          # Cutoff radius used for calculating the angular descriptor. Default is 5.0
     (Angst).
#ML_NHYP1 = 1           # Hyper parameter used for calculating the radial descriptor. Default is
    1.
#ML_NHYP2 = 4           # Hyper parameter used for calculating the angular descriptor. Default is
    4.
ML_MB=2000
ML_MCONF= 1600
```

D Lithium nitride (individual) INCAR

```
SYSTEM =Li3N
PREC = FAST
ALGO = Fast
SIGMA = 0.01 ; ISMEAR = 0
ENCUT =  350
NCORE = 4 ; KPAR =8
EDIFF = 1E-4 ; NELMIN = 6
ISYM = 0
LREAL = A
ISIF = 3
MDALGO=3                 # use Langevin thermostat
LANGEVIN_GAMMA = 1.0 1.0   # friction coef. for atomic DoFs for each species
LANGEVIN_GAMMA_L=3.0   # friction coef. for the lattice DoFs
PMASS=100                # mass for lattice DoFs
LATTICE_CONSTRAINTS = T T T   # fix x&y, release z lattice dynamics
PSTRESS=0.001            # P is set at 0.001 KB.
POTIM = 1.5 #2.50
IBRION = 0
SMASS = 0
TEBEG = 0.00 #  750.00D0
TEEND = 0.00 #750.00D0
NSW = 1 #200000 #xx K/ps
NBLOCK = 1
NWRITE = 1
NELM = 100
ISPIN = 1
INIWAV = 1
IWAVPR = 1
ISTART = 0
LWAVE = .FALSE.
LCHARG = .FALSE.
#####################################
### MACHINE-LEARNING          ###
#####################################
### General parameters        ###
ML_LMLFF = .TRUE.  # Set as LMLFF_FF = .TRUE., when some machine-learning force field
    calculations are executed. Default is .FALSE.
ML_ISTART = 1       # Parameter to control restarting calculations. Set 0, 1 or 2.
                    # When you execute the training from the scratch, this needs to be set to
                        0.
                    # When you restart the training reading the previous ab initio data (ABCAR
                        file), this needs to be set to 1.
ML_RCUT1 = 6.0          # Cutoff radius used for calculating the radial descriptor. Default is
    RCUT2_FFM.
ML_RCUT2 = 5.0          # Cutoff radius used for calculating the angular descriptor. Default is 5.0
     (Angst).
#ML_NHYP1 = 1           # Hyper parameter used for calculating the radial descriptor. Default is
    1.
#ML_NHYP2 = 4           # Hyper parameter used for calculating the angular descriptor. Default is
    4.
ML_MB=4000
ML_MCONF= 1200
ML_SCLC_CTIFOR = 0.0000001
```

E Diamond (truncated) input.nn

```
###############################################################################
# GENERAL NNP SETTINGS
###############################################################################
number_of_elements              1               # Number of elements.
elements                        C               # Specification of elements.
cutoff_type                     1 0.0           # Cutoff type (optional argument: shift parameter alpha).
scale_symmetry_functions_sigma                  # Scale all symmetry functions with sigma.
scale_min_short                 0.0             # Minimum value for scaling.
scale_max_short                 1.0             # Maximum value for scaling.
global_hidden_layers_short      2               # Number of hidden layers.
global_nodes_short              10 10           # Number of nodes in each hidden layer.
global_activation_short         p p l           # Activation function for each hidden layer and output layer.

###############################################################################
# ADDITIONAL SETTINGS FOR DATASET TOOLS
###############################################################################
use_short_forces                                # Use forces.
random_seed                     1235813         # Random number generator seed.

###############################################################################
# ADDITIONAL SETTINGS FOR TRAINING
###############################################################################
epochs                          10              # Number of training epochs.
normalize_data_set              force           # Normalize data set prior to training (ref = via ref. data, force = v
updater_type                    1               # Weight update method (0 = Gradient Descent, 1 = Kalman filter).
parallel_mode                   0               # Training parallelization used (0 = Parallel (rank 0 update), 1 = Pa
jacobian_mode                   1               # Jacobian computation mode (0 = Summation to single gradient, 1 = Per
update_strategy                 0               # Update strategy (0 = Combined, 1 = Per-element).
selection_mode                  2               # Update candidate selection mode (0 = Random, 1 = Sort, 2 = Threshold
task_batch_size_energy          1               # Number of energy update candidates prepared per task for each update
task_batch_size_force           1               # Number of force update candidates prepared per task for each update
memorize_symfunc_results                        # Keep symmetry function results in memory.
test_fraction                   0.05            # Fraction of structures kept for testing.
force_weight                    100.0           # Weight of force updates relative to energy updates.
short_energy_fraction           1.000           # Fraction of energy updates per epoch.
force_energy_ratio              3.0             # Specifies ratio between force and energy updates (ratio = updates_fo
short_energy_error_threshold    0.00            # RMSE threshold for energy update candidates.
short_force_error_threshold     1.00            # RMSE threshold for force update candidates.
rmse_threshold_trials           3               # Maximum number of RMSE threshold trials.
weights_min                     -1.0            # Minimum value for initial random weights.
weights_max                     1.0             # Maximum value for initial random weights.
main_error_metric               RMSEpa          # Main error metric for screen output (RMSEpa/RMSE/MAEpa/MAE).
write_trainpoints               1               # Write energy comparison every this many epochs.
write_trainforces               1               # Write force comparison every this many epochs.
write_weights_epoch             1               # Write weights every this many epochs.
write_neuronstats               1               # Write neuron statistics every this many epochs.
write_trainlog                                  # Write training log file.
###############################
# KALMAN FILTER (STANDARD) #
###############################
kalman_type                     0               # Kalman filter type (0 = Standard, 1 = Fading memory).
kalman_epsilon                  1.0E-2          # General Kalman filter parameter epsilon (sigmoidal: 0.01, linear: 0
kalman_q0                       0.01            # General Kalman filter parameter q0 ("large").
kalman_qtau                     2.302           # General Kalman filter parameter qtau (2.302 => 1 order of magnitude
kalman_qmin                     1.0E-6          # General Kalman filter parameter qmin (typ. 1.0E-6).
kalman_eta                      0.01            # Standard Kalman filter parameter eta (0.001-1.0).
kalman_etatau                   2.302           # Standard Kalman filter parameter etatau (2.302 => 1 order of magnitu
kalman_etamax                   1.0             # Standard Kalman filter parameter etamax (1.0+).
```

F Lithium nitride (truncated) input.nn

```
################################################################################
# GENERAL NNP SETTINGS
################################################################################
number_of_elements                2                 # Number of elements.
elements                          Li N              # Specification of elements.
cutoff_type                       1 0.0             # Cutoff type (optional argument: shift parameter alpha).
scale_symmetry_functions_sigma                      # Scale all symmetry functions with sigma.
scale_min_short                   0.0               # Minimum value for scaling.
scale_max_short                   1.0               # Maximum value for scaling.
global_hidden_layers_short        2                 # Number of hidden layers.
global_nodes_short                40 20             # Number of nodes in each hidden layer.
global_activation_short           p p l             # Activation function for each hidden layer and output layer.

################################################################################
# ADDITIONAL SETTINGS FOR DATASET TOOLS
################################################################################
use_short_forces                                    # Use forces.
random_seed                       1235813           # Random number generator seed.

################################################################################
# ADDITIONAL SETTINGS FOR TRAINING
################################################################################
epochs                            10                # Number of training epochs.
normalize_data_set                force             # Normalize data set prior to training (ref = via ref. data, force = v
updater_type                      1                 # Weight update method (0 = Gradient Descent, 1 = Kalman filter).
parallel_mode                     0                 # Training parallelization used (0 = Parallel (rank 0 update), 1 = Pa
jacobian_mode                     1                 # Jacobian computation mode (0 = Summation to single gradient, 1 = Per
update_strategy                   0                 # Update strategy (0 = Combined, 1 = Per-element).
selection_mode                    2                 # Update candidate selection mode (0 = Random, 1 = Sort, 2 = Threshold
task_batch_size_energy            1                 # Number of energy update candidates prepared per task for each update
task_batch_size_force             1                 # Number of force update candidates prepared per task for each update
memorize_symfunc_results                            # Keep symmetry function results in memory.
test_fraction                     0.05              # Fraction of structures kept for testing.
force_weight                      20.0              # Weight of force updates relative to energy updates.
short_energy_fraction             1.000             # Fraction of energy updates per epoch.
force_energy_ratio                3.0               # Specifies ratio between force and energy updates (ratio = updates_fo
short_energy_error_threshold      0.00              # RMSE threshold for energy update candidates.
short_force_error_threshold       1.00              # RMSE threshold for force update candidates.
rmse_threshold_trials             3                 # Maximum number of RMSE threshold trials.
weights_min                       -1.0              # Minimum value for initial random weights.
weights_max                       1.0               # Maximum value for initial random weights.
main_error_metric                 RMSEpa            # Main error metric for screen output (RMSEpa/RMSE/MAEpa/MAE).
write_trainpoints                 1                 # Write energy comparison every this many epochs.
write_trainforces                 1                 # Write force comparison every this many epochs.
write_weights_epoch               1                 # Write weights every this many epochs.
write_neuronstats                 1                 # Write neuron statistics every this many epochs.
write_trainlog                                      # Write training log file.
###############################
# KALMAN FILTER (STANDARD) #
###############################
kalman_type                       0                 # Kalman filter type (0 = Standard, 1 = Fading memory).
kalman_epsilon                    1.0E-2            # General Kalman filter parameter epsilon (sigmoidal: 0.01, linear: 0
kalman_q0                         0.01              # General Kalman filter parameter q0 ("large").
kalman_qtau                       2.302             # General Kalman filter parameter qtau (2.302 => 1 order of magnitude
kalman_qmin                       1.0E-6            # General Kalman filter parameter qmin (typ. 1.0E-6).
kalman_eta                        0.01              # Standard Kalman filter parameter eta (0.001-1.0).
kalman_etatau                     2.302             # Standard Kalman filter parameter etatau (2.302 => 1 order of magnitu
kalman_etamax                     1.0               # Standard Kalman filter parameter etamax (1.0+).
```

G Diamond in.lmp

```
######## Variables ########
clear
variable dt          equal 0.001
variable num_steps   equal 15000
variable start_temp  equal 100
variable end_temp    equal 3000
variable start_pressure equal 0
variable end_pressure   equal 0

#variable lattice_parameter equals 3.567
variable input_structure string "in.data"
variable input_nnp       string "../../training/diamond"

variable thermostat_seed  equal 223471
variable temperature_seed equal 987428

variable run_time equal "v_dt * v_num_steps"

######## Setup ########
units        metal
dimension    3
boundary     p p p
atom_style   atomic

# create box
#lattice      fcc ${lattice_parameter}
#region       entire_region block 0 10 0 10 0 10
#create_box   1 entire_region

# create atoms
#lattice       fcc ${lattice_parameter} orient x 1 0 0 y 0 1 0 z 0 0 1
#create_atoms  1 region entire_region
read_data ${input_structure}
mass 1 12.0107

replicate 1 1 1

######## Potential ########
neigh_modify one 4000

# cflength 1.8897261328 cfenergy 0.0367493254
pair_style nnp dir ${input_nnp} showew no showewsum 100 maxew 100000 resetew yes
pair_coeff * * 6.01

# computes
#compute peperatom all pe/atom

######## Sim ########
reset_timestep 0
timestep ${dt}

# init temp
variable twice_start_temp equal "2.0 * v_start_temp"
velocity all create ${twice_start_temp} ${temperature_seed}

# fixes
fix 1 all npt temp ${start_temp} ${end_temp} $(100.0*dt) aniso ${start_pressure} ${end_pressure} $(1000.0*dt) drag 1
#fix 1 all nve
#fix 2 all langevin ${start_temp} ${end_temp} 0.5 ${thermostat_seed} zero yes
#fix 1 all nvt temp ${start_temp} ${end_temp} $(100.0*dt)

# output
thermo 100
thermo_style custom step temp pe lx ly lz

dump 1 all custom 5 dump/dump.* id element x y z fx fy fz
dump_modify 1 element C

print "Running for ${run_time}ps"
run ${num_steps}
print "Done"
```

H Lithium nitride (150 K to 400 K) in.lmp

```
######## Variables ########
clear
variable dt          equal 0.001
variable num_steps   equal 100000
variable start_temp  equal 150
variable end_temp    equal 400
variable start_pressure equal 0
variable end_pressure   equal 0

variable input_structure string "in.data"
variable input_nnp       string "../../training/Li3N-full"

variable thermostat_seed  equal 926542
variable temperature_seed equal 432198

variable run_time equal "v_dt * v_num_steps"

######## Setup ########
units        metal
dimension    3
boundary     p p p
atom_style   atomic

# create atoms
read_data ${input_structure}
mass 1 6.941
mass 2 14.0067

replicate 1 1 1

######## Potential ########
neigh_modify one 4000

pair_style nnp dir ${input_nnp} showew no showewsum 100 maxew 100000 resetew yes
pair_coeff * * 6.01

# computes
group element_1 type 1
group element_2 type 2
compute msd_1 element_1 msd
compute msd_2 element_2 msd

######## Sim ########
reset_timestep 0
timestep ${dt}

# init temp
variable twice_start_temp equal "2.0 * v_start_temp"
velocity all create ${twice_start_temp} ${temperature_seed}

# fixes
fix 1 all npt temp ${start_temp} ${end_temp} $(100.0*dt) aniso ${start_pressure} ${end_pressure} $(1000.0*dt) drag 1
#fix 1 all nve
#fix 2 all langevin ${start_temp} ${end_temp} 0.5 ${thermostat_seed} zero yes
#fix 1 all nvt temp ${start_temp} ${end_temp} $(100.0*dt)

# output
thermo 10
thermo_style custom step temp pe lx ly lz c_msd_1[1] c_msd_1[2] c_msd_1[3] c_msd_1[4] c_msd_2[1] c_msd_2[2] c_msd_2[3

dump 1 all custom 100 dump/dump.* id element x y z fx fy fz
dump_modify 1 element Li N

print "Running for ${run_time}ps"
run ${num_steps}
print "Done"
```

I Lithium nitride (400 K to 800 K) in.lmp

```
######## Variables ########
clear
variable dt          equal 0.001
variable num_steps   equal 50000
variable start_temp  equal 400
variable end_temp    equal 800
variable start_pressure equal 0
variable end_pressure   equal 0

variable input_structure string "in.data"
variable input_nnp       string "../../training/Li3N-750K-full"

variable thermostat_seed  equal 926542
variable temperature_seed equal 432198

variable run_time equal "v_dt * v_num_steps"

######## Setup ########
units        metal
dimension    3
boundary     p p p
atom_style   atomic

# create atoms
read_data ${input_structure}
mass 1 6.941
mass 2 14.0067

replicate 1 1 1

######## Potential ########
neigh_modify one 4000

pair_style nnp dir ${input_nnp} showew no showewsum 100 maxew 100000 resetew yes
pair_coeff * * 6.01

# computes
group element_1 type 1
group element_2 type 2
compute msd_1 element_1 msd
compute msd_2 element_2 msd

######## Sim ########
reset_timestep 0
timestep ${dt}

# init temp
variable twice_start_temp equal "2.0 * v_start_temp"
velocity all create ${twice_start_temp} ${temperature_seed}

# fixes
fix 1 all npt temp ${start_temp} ${end_temp} $(100.0*dt) aniso ${start_pressure} ${end_pressure} $(1000.0*dt) drag 1
#fix 1 all nve
#fix 2 all langevin ${start_temp} ${end_temp} 0.5 ${thermostat_seed} zero yes
#fix 1 all nvt temp ${start_temp} ${end_temp} $(100.0*dt)

# output
thermo 10
thermo_style custom step temp pe lx ly lz c_msd_1[1] c_msd_1[2] c_msd_1[3] c_msd_1[4] c_msd_2[1] c_msd_2[2] c_msd_2[3

dump 1 all custom 100 dump/dump.* id element x y z fx fy fz
dump_modify 1 element Li N

print "Running for ${run_time}ps"
run ${num_steps}
print "Done"
```

J Diamond symmetry functions

Type	r_l	r_c	θ_l	θ_c
23	-1.0	1.0		
23	-0.5	1.6		
23	0.0	2.1		
23	0.5	2.7		
23	1.0	3.2		
23	1.6	3.8		
23	2.1	4.3		
23	2.7	4.9		
23	3.2	5.4		
23	3.8	5.9		
25	-3.0	3.0		
25	-1.5	4.5		
25	0.0	6.0		
25			135	225
25			-45	45
25			45	135
25			0	90
25			90	180

K Lithium nitride symmetry functions

Type	r_l	r_c	θ_l	θ_c
23	-1.0	1.0		
23	-0.5	1.6		
23	0.0	2.1		
23	0.5	2.7		
23	1.0	3.2		
23	1.6	3.8		
23	2.1	4.3		
23	2.7	4.9		
23	3.2	5.4		
23	3.8	5.9		
25	-4.0	4.0		
25	-2.0	6.0		
25			-20	2
25			0	4
25			2	60
25			4	80
25			6	100
25			8	120
25			1	140
25			1	160
25			1	180
25			1	200

(a) $10 + 2 \cdot 10$

Type	r_l	r_c	θ_l	θ_c
23	-0.5	0.5		
23	-0.2	0.8		
23	0.0	1.1		
23	0.2	1.4		
23	0.5	1.7		
23	0.8	1.9		
23	1.1	2.2		
23	1.4	2.5		
23	1.7	2.8		
23	1.9	3.1		
23	2.2	3.4		
23	2.5	3.7		
23	2.8	4.0		
23	3.1	4.2		
23	3.4	4.5		
23	3.7	4.8		
23	4.0	5.1		
23	4.2	5.4		
23	4.5	5.7		
23	4.8	5.9		
25	-4.0	4.0		
25	-2.0	6.0		
25			-45	45
25			0	90
25			45	136
25			90	180
25			135	225

(b) $20 + 2 \cdot 5$

Type	r_l	r_c	θ_l	θ_c
23	-0.5	0.5		
23	-0.2	0.8		
23	0.0	1.1		
23	0.2	1.4		
23	0.5	1.7		
23	0.8	1.9		
23	1.1	2.2		
23	1.4	2.5		
23	1.7	2.8		
23	1.9	3.1		
23	2.2	3.4		
23	2.5	3.7		
23	2.8	4.0		
23	3.1	4.2		
23	3.4	4.5		
23	3.7	4.8		
23	4.0	5.1		
23	4.2	5.4		
23	4.5	5.7		
23	4.8	5.9		
25	-2.4	2.4		
25	-1.2	3.5		
25	0.0	4.8		
25	1.1	6.0		
25			-20	20
25			0	40
25			20	60
25			40	80
25			60	100
25			80	120
25			100	140
25			120	160
25			140	180
25			160	200

(a) $20 + 4 \cdot 10$

Type	r_l	r_c	θ_l	θ_c
23	-0.3	0.3		
23	-0.1	0.5		
23	0.0	0.7		
23	0.1	0.9		
23	0.3	1.1		
23	0.5	1.3		
23	0.7	1.5		
23	0.9	1.7		
23	1.1	1.9		
23	1.3	2.1		
23	1.5	2.3		
23	1.7	2.5		
23	1.9	2.7		
23	2.1	2.9		
23	2.3	3.0		
23	2.5	3.2		
23	2.7	3.4		
23	2.9	3.6		
23	3.0	3.8		
23	3.2	4.0		
23	3.4	4.2		
23	3.6	4.4		
23	3.8	4.6		
23	4.0	4.8		
23	4.2	5.0		
23	4.4	5.2		
23	4.6	5.4		
23	4.8	5.6		
23	5.0	5.8		
23	5.2	5.9		
25	-3.0	3.0		
25	-1.5	4.5		
25	0.0	6.0		
25			-20	20
25			0	40
25			20	60
25			40	80
25			60	100
25			80	120
25			100	140
25			120	160
25			140	180
25			160	200

(b) $30 + 3 \cdot 10$

Type	r_l	r_c	θ_l	θ_c
23	-0.2	0.2		
23	-0.1	0.4		
23	0.0	0.5		
23	0.1	0.7		
23	0.2	0.8		
23	0.4	1.0		
23	0.5	1.1		
23	0.7	1.3		
23	0.8	1.4		
23	1.0	1.6		
23	1.1	1.7		
23	1.3	1.9		
23	1.4	2.0		
23	1.6	2.1		
23	1.7	2.3		
23	1.9	2.4		
23	2.0	2.6		
23	2.1	2.7		
23	2.3	2.9		
23	2.4	3.0		
23	2.6	3.2		
23	2.7	3.3		
23	2.9	3.5		
23	3.0	3.6		
23	3.2	3.8		
23	3.3	3.9		
23	3.5	4.0		
23	3.6	4.2		
23	3.8	4.3		
23	3.9	4.5		
23	4.0	4.6		
23	4.2	4.8		
23	4.3	4.9		
23	4.5	5.1		
23	4.6	5.2		
23	4.8	5.4		
23	4.9	5.5		
23	5.1	5.7		
23	5.2	5.8		
23	5.4	6.0		
25	-2.4	2.4		
25	-1.2	3.5		
25	0.0	4.8		
25	1.1	6.0		
25			-45	45
25			0	90
25			45	136
25			90	180
25			135	225

(a) $40 + 4 \cdot 5$

Type	r_l	r_c	θ_l	θ_c
23	-0.3	0.3		
23	-0.1	0.5		
23	0.0	0.7		
23	0.1	0.9		
23	0.3	1.1		
23	0.5	1.3		
23	0.7	1.5		
23	0.9	1.7		
23	1.1	1.9		
23	1.3	2.1		
23	1.5	2.3		
23	1.7	2.5		
23	1.9	2.7		
23	2.1	2.9		
23	2.3	3.0		
23	2.5	3.2		
23	2.7	3.4		
23	2.9	3.6		
23	3.0	3.8		
23	3.2	4.0		
23	3.4	4.2		
23	3.6	4.4		
23	3.8	4.6		
23	4.0	4.8		
23	4.2	5.0		
23	4.4	5.2		
23	4.6	5.4		
23	4.8	5.6		
23	5.0	5.8		
23	5.2	5.9		
25	-2.4	2.4		
25	-1.2	3.5		
25	0.0	4.8		
25	1.1	6.0		
25			-20	20
25			0	40
25			20	60
25			40	80
25			60	100
25			80	120
25			100	140
25			120	160
25			140	180
25			160	200
25			-90	90
25			90	270
25			0	180
25			0	90
25			90	180

(b) $30 + 4 \cdot 15$

Type	r_l	r_c	θ_l	θ_c
23	-0.2	0.2		
23	-0.1	0.3		
23	0.0	0.4		
23	0.1	0.5		
23	0.2	0.7		
23	0.3	0.8		
23	0.4	0.9		
23	0.5	1.0		
23	0.7	1.1		
23	0.8	1.2		
23	0.9	1.4		
23	1.0	1.5		
23	1.1	1.6		
23	1.2	1.7		
23	1.4	1.8		
23	1.5	2.0		
23	1.6	2.1		
23	1.7	2.2		
23	1.8	2.3		
23	2.0	2.4		
23	2.1	2.5		
23	2.2	2.7		
23	2.3	2.8		
23	2.4	2.9		
23	2.5	3.0		
23	2.7	3.1		
23	2.8	3.2		
23	2.9	3.4		
23	3.0	3.5		
23	3.1	3.6		
23	3.2	3.7		
23	3.4	3.8		
23	3.5	4.0		
23	3.6	4.1		
23	3.7	4.2		
23	3.8	4.3		
23	4.0	4.4		
23	4.1	4.5		
23	4.2	4.7		
23	4.3	4.8		
23	4.4	4.9		
23	4.5	5.0		
23	4.7	5.1		
23	4.8	5.2		
23	4.9	5.4		
23	5.0	5.5		
23	5.1	5.6		
23	5.2	5.7		
23	5.4	5.8		
23	5.5	6.0		
25	-2.4	2.4		
25	-1.2	3.5		
25	0.0	4.8		
25	1.1	6.0		
25			-20	20
25			0	40
25			20	60
25			40	80
25			60	100
25			80	120
25			100	140
25			120	160
25			140	180
25			160	200

(a) $50 + 4 \cdot 10$