Freie Universität Berlin

A fast implementation of deep neural-network potentials for molecular dynamics simulations of alloys

by

Sandip Kumar Sah

Master Thesis in Computational Science

Submission: 09 January 2026 Supervisor: Prof. Dr. Felix Höfling

Freie Universität Berlin | Department of Mathematics and Computer Science
Institute of Mathematics

Statutory Declaration

Family Name, Given/First Name	Sah, Sandip Kumar
Matriculation number	5589263
Kind of thesis submitted	Master Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

.....

Date, Signature

Abstract

A concise 200–300 word summary of:

- Motivation (accelerating atomistic simulations using neural network potentials)
- Goal (integrating DeepMD potential calculation into HALMD)
- Method (extracting weights, replicating inference in C++/CUDA)
- Key results and performance (accuracy vs. speed trade-off)
- Conclusions

Contents

1 Introduction

1.1 Motivation

Molecular dynamics (MD) simulations play a central role in materials science by providing atomistic insight into the behavior of complex systems. Their predictive power depends on the interatomic potential used to approximate the underlying potential energy surface (PES). Classical empirical potentials are efficient but often too rigid to describe complex bonding environments, whereas *ab initio* methods offer high accuracy at prohibitive computational cost. Machine-learned interatomic potentials—particularly the Deep Potential Molecular Dynamics (DeepMD) framework—bridge this gap by achieving near–quantum mechanical accuracy with classical-MD performance.

The work by Andrés Cruz, titled *"Deep Neural Networks Potentials for Scalable Molecular Dynamics Simulations on Accelerator Hardware"* [1], represents an important step toward integrating Deep Potential models into the high-performance GPU-accelerated simulation engine HALMD. His work reconstructs the DeePMD-kit inference pipeline inside HALMD, extracts network weights from a trained TensorFlow model, and validates energy and force predictions for a *single-species Copper system*. In particular, his implementation focuses on the *two-body embedding DeepPot-SE descriptor* and reproduces the filter and fitting networks *only for a monoatomic system*, making it suitable for single metal.

However, the implementation in [1] remains limited to the simplest case of DeepMD-v2 architectures. It does not include multi-species support or the full range of descriptor and network features available in DeepMD-kit v2.

Additionally, several intermediate steps present in the full DeepMD-v2 computational graph—such as normalization layers, ghost body extension for periodicity in boundary condition, species-wise descriptor partitioning, and certain derivative pathways—were simplified or omitted in the previous implementation. Cruz's work successfully established a working single-species DeepMD integration within HALMD.

This work builds directly on the work of Cruz [1] and advances it substantially. The primary contributions of the present work are:

1. Generalizing the HALMD Deep Potential integration to *multi-species, multi-body DeepMD-v2 models*, enabling simulations of binary and multicomponent alloy systems.
2. Reconstructing descriptor and fitting networks for the *full multi-species DeepPot-SE architecture*, including multi-body descriptor terms.
3. Improving the *accuracy* of the HALMD implementation by adding several computational steps missing in the previous work, such as proper cutoff normalization, multi-species descriptor algebra, and the full force backpropagation pipeline.

Through these extensions, the present thesis transforms HALMD from supporting a prototype single-component DeepMD potential into a *fully general, high-accuracy, multi-species DeepMD-v2 engine*. This significantly broadens HALMD's applicability, enabling large-scale molecular dynamics simulations of technologically relevant multicomponent materials at near–quantum mechanical accuracy.

1.2 Objectives and Scope

The objective of this thesis is to extend and generalize the Deep Potential (DP) implementation in HALMD beyond the single-species, two-body descriptor developed by Cruz [1]. While Cruz's work successfully demonstrated that HALMD can evaluate a DeePMD-v2 model for a monoatomic copper system, several components required for full multi-species DeepMD-v2 inference were missing. Most notably, the previous implementation did not include (i) the periodic coordinate extension and neighbor-list construction used in DeepMD [2], (ii) normalization and scaling layers defined in the DP-v2 framework [3], (iii) species-dependent descriptors, or (iv) descriptor and filter weights that depend simultaneously on the central and neighbor species, as introduced in the multi-species DeepPot-SE descriptor [3].

This thesis addresses these limitations by implementing the complete multi-species DeepMD-v2 inference pipeline, including accurate periodic handling of atomic environments. Specifically, the thesis pursues the following objectives:

1. **Implement coordinate normalization, ghost-cell extension, and multi-type neighbor list construction.** The previous implementation did not include the canonical DeePMD preprocessing steps for periodic systems—wrapping coordinates into the primary simulation cell, generating ghost atoms to cover the cutoff radius, and constructing species-grouped neighbor lists—as described in the DeePMD methodology [2, 3]. Implementing these steps ensures that HALMD reproduces the correct local environments required by the DeepPot-SE descriptors under periodic boundary conditions.

2. **Generalize the descriptor pipeline to multi-species systems.** This includes implementing species-dependent neighbor counts $N_c(a)$, species-aware embedding matrices G_i, and the full multi-species DeepPot-SE (se_e2_a) descriptor introduced in DP-v2 [3].

3. **Implement species-dependent filter networks.** Extend the single-species embedding and filter networks used in [1] to support arbitrary numbers of atomic types, following the species-indexed filter network formulation defined in the DP-v2 architecture [3].

4. **Reproduce the full DeepMD-v2 inference procedure with higher accuracy.** Incorporate several computational steps omitted in previous work, including normalization layers, descriptor scaling operations, smooth cutoff functions, and full force backpropagation through descriptor derivatives, consistent with the DeepMD-v2 formulation [3, 2].

5. **Validate energy and force predictions for multi-component systems.** Compare results from HALMD against the DeePMD-kit reference implementation for multi-species models and quantify numerical deviations, following standard validation procedures established in DeepMD literature [2, 3].

Scope. This thesis focuses exclusively on the inference stage of DeepMD-v2, i.e., the computation of energies and forces from pre-trained models. Model training is outside the scope of this work. The implementation targets the multi-species DeepPot-SE descriptor and does not cover other descriptor families such as end-to-end or message-passing potentials. The work provides support for multi-species atomic systems relevant to alloys and chemically complex materials, while the underlying molecular dynamics algorithms

(integrators, thermostats, barostats) rely on HALMD's existing infrastructure.

2 Background

2.1 HALMD Software

The High-Accuracy Large-scale Molecular Dynamics (HALMD) package is a high-performance, open-source molecular dynamics (MD) simulation framework designed to study the microscopic dynamics of liquids, glasses, and other condensed matter systems. HALMD is built around a modular C++ architecture with a strong emphasis on numerical precision, extensibility, and efficient parallel computation on graphics processing units (GPUs) [4]. It provides a versatile platform for conducting large-scale classical molecular dynamics simulations involving millions of particles.

HALMD employs the *classical molecular dynamics* approach, in which atomic motion is governed by Newton's equations of motion,

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} = -\nabla_i U(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N),$$

where U is a predefined analytical potential describing interatomic interactions. The accuracy of a classical MD simulation depends on the fidelity of this potential. HALMD supports several built-in empirical pair potentials, such as Lennard–Jones, Gaussian Core, and Yukawa models, and additionally allows users to supply tabulated pair potentials. These empirical models do not involve quantum-mechanical calculations and thus differ from *ab initio* molecular dynamics (AIMD), where forces are computed directly from electronic structure methods.

One of HALMD's defining strengths is its high-performance GPU backend. The software employs NVIDIA's CUDA to accelerate computationally intensive tasks such as force evaluation, neighbor-list construction, and time integration [4], and more recent versions extend support to heterogeneous computing platforms through SYCL [5]. Spatial domain decomposition and cell binning enable efficient scaling on multi-GPU systems, achieving orders-of-magnitude speedups compared to CPU-only implementations. HALMD also supports mixed-precision arithmetic, selectively applying double precision to critical operations to ensure long-term numerical stability without compromising performance.

HALMD follows a modular design philosophy. Core components—integrators, interaction potentials, neighbor-list builders, and observables—are implemented as interchangeable modules configurable through a Lua scripting interface [6]. This modularity makes it straightforward to extend HALMD with new functionality, including custom potentials, analysis routines, and data exporters. Simulation data are stored in the H5MD format [7], an HDF5-based standard widely supported by analysis tools in computational physics and chemistry.

These characteristics make HALMD an ideal platform for integrating machine-learned interatomic potentials. Its GPU-accelerated, modular architecture provides the necessary infrastructure to replace conventional analytical potentials with neural-network-based force fields. In this thesis, HALMD serves as the host MD engine into which the Deep Potential Molecular Dynamics (DeepMD) model is embedded. The goal is to achieve *ab initio*-level accuracy within HALMD's efficient large-scale simulation framework by implementing a full multi-species DeepMD-v2 inference pipeline.

2.2 Neural Network Potentials

In recent years, machine learning techniques—particularly deep neural networks (DNNs)—have revolutionized the construction of interatomic potentials for molecular dynamics simulations. Traditional analytical potentials, while computationally efficient, are often limited in their ability to accurately capture many-body interactions and chemical complexities across diverse atomic environments. Neural Network Potentials (NNPs) overcome these limitations by learning the potential energy surface (PES) directly from *ab initio* reference data [8, 9].

In the NNP framework, a neural network is trained to approximate the mapping between an atomic configuration and its corresponding total energy and atomic forces. The total potential energy of a system is commonly expressed as a sum of atomic contributions [8]:

$$E = \sum_i E_i(\mathcal{R}_i),$$

where E_i denotes the atomic energy associated with atom i, and $\mathcal{R}_i$ represents its local environment within a cutoff radius. This decomposition ensures extensivity and locality and enables efficient scaling with system size.

Training a neural network potential involves minimizing a loss function that combines the errors in energies, forces, and optionally virials. Following the formulation used in DeePMD-kit [2, 3], the loss is written as

$$\mathcal{L} = p_E L_E + p_F L_F + p_V L_V, \tag{1}$$

where the terms are defined as

$$L_E = \frac{1}{N_E} \sum_{n=1}^{N_E} \left(E_n^{\mathrm{NN}} - E_n^{\mathrm{DFT}}\right)^2, \tag{2}$$

$$L_F = \frac{1}{3N_F} \sum_{n=1}^{N_F} \sum_{i=1}^{N_{\mathrm{atoms}}} \left\|\mathbf{F}_{i,n}^{\mathrm{NN}} - \mathbf{F}_{i,n}^{\mathrm{DFT}}\right\|^2, \tag{3}$$

$$L_V = \frac{1}{9N_V} \sum_{n=1}^{N_V} \left\|\mathbf{V}_n^{\mathrm{NN}} - \mathbf{V}_n^{\mathrm{DFT}}\right\|^2. \tag{4}$$

The factors $1/3$ and $1/9$ arise from averaging the force and virial errors over their three and nine Cartesian components, respectively, ensuring consistent normalization across all contributions to the loss.

During inference, atomic coordinates are transformed into symmetry-preserving descriptors that are invariant under translation, rotation, and permutation of atoms of the same type [10, 11]. These descriptors serve as input to the neural network, which predicts atomic energies and corresponding forces in real time, achieving *ab initio*-level accuracy at computational cost comparable to classical MD.

A variety of neural-network-based interatomic potentials have been proposed, including Behler–Parrinello networks [8], Gaussian Approximation Potentials (GAP) [12], SchNet [13], and the E(3)-equivariant NequIP model [14]. Among these, the Deep Potential Molecular Dynamics (DeepMD) framework has emerged as one of the most widely adopted and computationally efficient implementations due to its smooth descriptor formulation and native GPU acceleration. The following subsection focuses specifically on **DeepMD-kit version 2**, which forms the theoretical and computational foundation for the present work.

2.3 Deep Potential Molecular Dynamics (DeepMD)

This work focuses on **Deep Potential Molecular Dynamics version 2 (DeepMD-kit v2)**, the second-generation implementation of the Deep Potential framework. DeepMD-kit v2 represents a major advancement over the original DeepMD formulation [2], introducing improved descriptor smoothness, enhanced multi-species handling, and a refined software architecture that enables highly efficient GPU execution [3]. These features make DeepMD-kit v2 particularly suitable for modeling chemically complex systems such as binary alloys within HALMD.

Deep Potential Molecular Dynamics is a machine-learning approach that approximates interatomic interactions with near *ab initio* accuracy while retaining computational efficiency comparable to classical MD. A neural network is trained on quantum-mechanical reference data (typically from DFT or AIMD) to map atomic configurations to energies and forces. The total potential energy of the system is decomposed into atomic energy contributions [2, 3]:

$$E = \sum_i E_i(\mathcal{R}_i),$$

where E_i is the atomic energy of atom i, and $\mathcal{R}_i$ denotes its local environment within a cutoff radius r_c. This locality assumption yields linear scaling in the number of atoms and enables efficient parallelization on GPUs.

A defining element of DeepMD is the use of symmetry-preserving descriptors that are invariant under translations, rotations, and permutations of atoms of the same type. DeepMD-kit v2 employs the *Smooth Edition* (SE) descriptor family, provided in angular (`se_e2_a`) variants [3]. It is constructed from two matrices:

- the R**-matrix**, containing relative position vectors $R_{ij} = \mathbf{r}_j - \mathbf{r}_i$ for neighbors j within the cutoff radius;
- the G**-matrix**, a normalized representation incorporating angular information and many-body geometric correlations.

Distances $r_{ij} = |R_{ij}|$ are modulated by a smooth cutoff function to ensure continuity of energies and forces at the cutoff boundary [2].

The Deep Potential model in both DP-v1 and DP-v2 consists of two neural networks: an *embedding (filter) network* and a *fitting network* [2, 3].

1. **Embedding (Filter) Network.** This smaller feed-forward network transforms raw neighbor information into high-dimensional filter features. In DP-v2, a distinct embedding network is assigned to each atomic species, allowing the model to capture chemically specific interactions (e.g., Cu–Ag, Ni–Al) while preserving permutation invariance within each species [3]. The embedding network processes each neighbor independently.
2. **Fitting Network.** The outputs of the embedding networks are aggregated into a descriptor vector $\mathbf{D}_i$, which is passed to a larger fully connected fitting network that predicts the atomic energy E_i. Each species has its own fitting network parameters, enabling accurate modeling of multi-component materials.

Forces are obtained as the negative gradients of the total energy:

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{r}_i}.$$

During training, these derivatives are computed automatically through TensorFlow's back-propagation engine. During inference—which is the focus of this thesis—the gradients are obtained by applying the chain rule explicitly through the descriptor and neural-network layers, following the DP-v2 computational graph [3].

DeepMD-kit v2 introduces several enhancements that are essential for accurate multi-species modeling. These include species-dependent embedding networks, species-specific filter weights, descriptor normalization layers, and refined smooth cutoff schemes [3]. Together, these improvements enable DP-v2 to handle chemically diverse systems with greater smoothness, transferability, and numerical stability compared to its predecessor.

A trained DeepMD model is stored as a TensorFlow *frozen_model.pb* file along with an accompanying `input.json` file specifying descriptor types, cutoff radii, hyperparameters, and precision settings. During inference, atomic coordinates are transformed into descriptors, processed by species-specific embedding networks, and fed into the fitting network to produce atomic energies and forces.

In this thesis, DeepMD-kit v2 serves as the machine-learned potential that is fully integrated into HALMD. All parameter extraction, descriptor reconstruction, and neural-network inference follow the DP-v2 specification precisely. By embedding this inference process into HALMD's GPU-accelerated architecture, the present work enables large-scale simulations of multi-species systems, such as binary alloys, with near *ab initio* accuracy.

3 Methodology

3.1 Overview of Implementation

3.2 Model Parameter Extraction

3.2.1 Frozen Model Structure

3.2.2 Extraction Procedure

DeepMD-kit stores trained neural network potentials as a TensorFlow `frozen_model.pb` file, accompanied by an `input.json` file containing model hyperparameters. The `.pb` file encodes all numerical weights, biases, and auxiliary tensors in the form of TensorFlow computation graph nodes, while the JSON file specifies the descriptor type, cutoff radius, number of neurons per layer, activation functions, and species layout. Following the methodology of DeepMD-kit v1 and v2 [2, 3], this thesis extracts all necessary model parameters from these files and converts them into an HDF5 representation suitable for efficient inference inside HALMD.

The extraction process begins by loading the TensorFlow graph definition from the `.pb` file. All tensors of type `Const` are scanned, and those whose node names match descriptor or fitting-network layers are decoded using TensorFlow's low-level `tensor_util.MakeNdarray`. The DeepMD model contains one set of parameters per atomic species, and the species ordering in the output strictly follows the ordering defined in the DeepMD input configuration. For the work presented here, the descriptor type is always the angular Smooth Edition descriptor `se_e2_a`, which DeepMD-kit v2 uses for multi-species models requiring both radial and angular correlation encoding.

Descriptor Parameters. For each species, the descriptor section consists of a stack of fully connected layers with user-specified neuron counts, activation functions, and optional residual time-step connections. The extraction script identifies each descriptor layer via a naming pattern of the form

$$\texttt{filter\_type\_}\langle s\rangle/\texttt{matrix\_}\langle k\rangle\texttt{\_}\langle t_n\rangle, \qquad \texttt{filter\_type\_}\langle s\rangle/\texttt{bias\_}\langle k\rangle\texttt{\_}\langle t_n\rangle,$$

where s indexes the species, k is the layer index, and t_n enumerates neighbor-type channels. For each layer, the script records:

- weight matrices,
- bias vectors,
- number of neurons,
- activation function (typically `tanh` or `linear` in the present work),
- the presence of a residual network branch.

DeepMD-v2 allows descriptor layers to use residual updates when `resnet_dt=true`. In that case, a per-layer time-step tensor is extracted and marked in the output structure, though the use of residual updates depends on the model's training configuration.

Embedding (Filter) Network. In the DeepMD architecture, the descriptor network is conceptually equivalent to the "filter" or embedding network described in [2, 3]. Each species has its own filter-network parameters to account for species-dependent geometric correlations. The extracted filter-network weights are reshaped into a row-major layout compatible with HALMD's GPU evaluation kernels.

Fitting Network. For each species, the fitting network (also called the main network) predicts the atomic energy contribution E_i. The extraction process retrieves:

- all intermediate fitting layers,
- species-dependent activation functions,
- weight and bias tensors for each layer,
- the species-specific atomic energy bias term `bias_atom_e`,
- the parameters of the final output layer.

The fitting-network layers are identified using a template of the form

$$\texttt{layer\_LL\_type\_} < k >_< s > /\texttt{matrix}, \qquad \texttt{layer\_LL\_type\_} < k >_< s > /\texttt{bias},$$

and similarly for the final layer `final_layer_type_<s>`. The final layer uses a fixed activation function, typically `linear`, consistent with the DeepMD energy formulation.

Normalization Parameters. DeepMD-v2 introduces descriptor normalization tensors `t_avg` and `t_std`, which are required to maintain numerical stability and descriptor smoothness. These tensors are extracted from the nodes

$$\texttt{descrpt\_attr/t\_avg}, \qquad \texttt{descrpt\_attr/t\_std},$$

and stored in the HDF5 output so that HALMD can apply the same normalization as the original DeepMD model.

Organization and Output Format. All extracted parameters are written into a structured HDF5 file using a hierarchical layout:

- global descriptor constants (cutoff radii, smoothing radii, normalization tensors),
- per-species descriptor network parameters,
- per-species fitting network parameters.

This structure mirrors the multi-species design of DeepMD-kit v2 and makes the extracted parameters directly usable by HALMD for inference. Unlike the single-species extraction used in the previous HALMD implementation [1], the present work extracts and stores fully species-resolved descriptor and fitting-network data, which are required for accurate multi-species DeepMD-v2 inference.

The resulting HDF5 file serves as the unified model representation for HALMD. During the simulation, HALMD loads this file, reconstructs the descriptor and neural networks using GPU-optimized data structures, and performs inference without relying on TensorFlow. This enables the Deep Potential model to be evaluated natively within HALMD's simulation loop with high performance and full multi-species support.

3.3 Coordinate System Extension

A central contribution of this thesis is the redesign of HALMD's environment–construction pipeline so that it follows the exact conventions required by DeepMD-v2. The earlier implementation by Cruz [1] relied on HALMD's built-in periodic boundary handling and neighbour list, which correctly satisfy the minimum-image convention used in classical MD [4]. However, this approach provides only the displacements between atoms and does not reproduce the full periodic environment expected by the Deep Potential (DP) descriptor pipeline. DeepMD-v2 uses a stricter definition of the local atomic environment, requiring explicit coordinate wrapping, ghost-cell expansion, species-aware neighbour grouping, and descriptor normalization [2, 3]. These steps were not included in the previous HALMD implementation and form the foundation of the extended environment-construction procedure in this thesis.

The new DeePMD-style environment constructor introduced in this work consists of the following components:

1. Explicit coordinate wrapping. DeepMD assumes that all atomic coordinates are wrapped into the primary simulation cell before any descriptor computation. Although HALMD internally tracks particle images, its neighbour-displacement logic does not enforce wrapped coordinates. The new implementation applies explicit wrapping:

$$\tilde{r}_{i\alpha} = r_{i\alpha} - L_\alpha \left\lfloor \frac{r_{i\alpha}}{L_\alpha} \right\rfloor ,$$

ensuring strict consistency with DeepMD's periodic coordinate convention.

2. Ghost-cell periodic extension. DeepMD constructs local atomic environments by tiling the simulation box with enough periodic images to completely cover the descriptor cutoff radius r_c. This requirement is stricter than HALMD's neighbour list, which only

returns the nearest periodic image of each atom. To satisfy the descriptor definition, this thesis implements full ghost-cell extension:

$$\mathbf{r}_i^{(\mathbf{s})} = \mathbf{r}_i + s_x L_x \mathbf{e}_x + s_y L_y \mathbf{e}_y + s_z L_z \mathbf{e}_z, \quad s_\alpha \in [-n_{\mathrm{buff},\alpha}, n_{\mathrm{buff},\alpha}],$$

where $n_{\mathrm{buff},\alpha} = \lceil r_c / L_\alpha \rceil$. This guarantees that atoms near the boundary have complete environments identical to those produced by DeepMD-kit.

3. Species-aware neighbour grouping via the `sel` vector. DeepMD-v2 requires that neighbours be grouped first by species and only then sorted by distance, with the number of neighbours per species fixed by the descriptor configuration:

$$\texttt{sel} = [N_c(a_0), N_c(a_1), \dots].$$

HALMD's native neighbour list is species-blind and distance-only. The extended implementation introduced here constructs:

$$\text{neighbors\_by\_type}[a] = \{\, j \mid t_j = a,\ r_{ij} < r_c \,\},$$

sorts each group by distance, and fills descriptor rows in the exact ordering expected by the DP-v2 architecture. This step is essential for multi-species DeepPot-SE descriptors.

4. Environment-matrix construction interface (without descriptor math). In the previous HALMD implementation, the environment was computed directly using the neighbour list and minimum-image displacements. In this thesis, environment construction is redesigned to operate on the wrapped and ghost-extended coordinates, and on the species-grouped neighbour lists described above. The computation of the descriptor tensors themselves (such as the R and G matrices) is performed in a later stage and is documented separately in Section 3.4.

5. Descriptor normalization using `t_avg` and `t_std`. DeepMD-v2 applies per-feature normalization using statistics computed during training:

$$X' = \frac{X - t_{\mathrm{avg}}}{t_{\mathrm{std}}},$$

to improve numerical stability. Cruz's implementation did not include this step. The extended HALMD implementation integrates normalization directly into the environment-construction pipeline, ensuring full compatibility with the DP-v2 descriptor specification.

Summary of improvements. Relative to the previous work, the environment construction introduced in this thesis adds:

- strict DeePMD-style coordinate wrapping,
- full periodic ghost-cell expansion,
- species-aware neighbour grouping according to the DP `sel` configuration,
- a consistent environment-construction workflow aligned with DeepMD-v2,
- descriptor normalization compatible with training-time statistics.

These extensions form the necessary foundation for computing the R and G descriptor matrices (Section 3.4) and enable HALMD to accurately evaluate multi-species DeepMD-v2 models.

3.4 Computation of the R and G Matrices

The DeepPot-SE (DP-SE) descriptor is constructed in two stages. First, the geometric matrix R encodes neighbour geometry using inverse-distance–based quantities. Second, the embedding matrix G transforms these geometric inputs through species-dependent neural networks. Together, these matrices form the basis of the final DeepMD-v2 descriptor.

The implementation presented here reproduces the DeepMD-v2 specification exactly [2, 3] and corrects limitations of the earlier HALMD prototype [1].

3.4.1 R-Matrix Construction

The geometric matrix R encodes the relative positions of selected neighbours of each central atom and is designed to be invariant under translation, rotation, and permutation of atoms of the same species [2, 3]. Each neighbour contributes one row to the raw R-matrix.

Relative displacement and inverse-distance quantities. For a central atom i and neighbour j,

$$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i, \qquad r_{ij} = \|\mathbf{r}_{ij}\|.$$

DeepMD defines the raw geometric row as

$$R_{ij} = \left[s_{ij},\ \frac{x_{ij}}{r_{ij}^2},\ \frac{y_{ij}}{r_{ij}^2},\ \frac{z_{ij}}{r_{ij}^2} \right] f_{\mathrm{sw}}(r_{ij}),$$

where $s_{ij} = 1/r_{ij}$ and

$$\tilde{s}_{ij} = s_{ij} f_{\mathrm{sw}}(r_{ij})$$

is the cutoff-weighted inverse distance used in the descriptor.

The switching function $f_{\mathrm{sw}}(r)$ smoothly suppresses neighbour contributions near the cutoff:

$$f_{\mathrm{sw}}(r) = \begin{cases} 1, & r \le r_{\mathrm{sm}}, \\ x^3(-6x^2 + 15x - 10) + 1, & r_{\mathrm{sm}} < r < r_c, \\ 0, & r \ge r_c, \end{cases} \qquad x = \frac{r - r_{\mathrm{sm}}}{r_c - r_{\mathrm{sm}}}.$$

Species-aware neighbour selection. DeepMD-v2 requires that neighbours be:

1. grouped by species,
2. sorted by distance within each species group,
3. truncated according to the model's `sel` vector.

The previous HALMD implementation used a single flat neighbour list. The present work introduces complete species-aware ordering, ensuring perfect compatibility with the DeepMD-v2 descriptor format.

Periodic ghost-cell extension. DeepMD constructs environments using extended periodic images, not the minimum-image convention. This work reproduces the ghost-cell procedure described in Section 3.3, guaranteeing that $\mathbf{r}_{ij}$ is computed exactly as in DeepMD-kit.

Normalization. DeepMD-v2 applies component-wise normalization to each row of R using model parameters `t_avg` and `t_std`:

$$\hat{R}_{ij,\alpha} = \frac{R_{ij,\alpha} - (t_{\text{avg}})_\alpha}{(t_{\text{std}})_\alpha}.$$

In particular, the normalized inverse distance

$$\hat{s}_{ij} = \hat{R}_{ij,0}$$

is the scalar fed into the embedding networks.

Summary. The updated R-matrix construction introduces:

- species-aware neighbour selection,
- periodic ghost-cell extension,
- exact DP-SE switching behaviour,
- normalization using t_{avg} and t_{std}.

These improvements produce geometric descriptors fully consistent with DeepMD-v2.

3.4.2 G-Matrix (Embedding Matrix) Construction

The second stage of the descriptor constructs the embedding matrix G by passing each normalized inverse distance $\hat{s}_{ij}$ through a small neural network known as a *filter network*. The output is a vector

$$G_{ij} \in \mathbb{R}^M,$$

where M is the embedding dimension.

Central–neighbour species dependence. For a central atom of species a and neighbour of species b, DeepMD-v2 defines a distinct embedding network:

$$G_{ij} = N_\theta^{(a,b)}(\hat{s}_{ij}).$$

This species-pair dependence is essential for modelling cross-species interactions (e.g., Cu–Ni, Ni–Al). The previous HALMD implementation [1] used only a single network per central species:

$$G_{ij} = N_\theta^{(a)}(\hat{s}_{ij}),$$

which cannot represent multi-component models.

The present work extends HALMD to store and evaluate:

$$\texttt{neural\_networks[a][b]} \equiv N_\theta^{(a,b)},$$

thus implementing the full DeepMD-v2 filter-network structure.

Runtime network selection. During environment construction, the species-ordered neighbour list assigns a neighbour type index b to each row j. Thus the embedding evaluation becomes:

$$G_{ij} = N_\theta^{(a,b_{(j)})}(\hat{s}_{ij})\,.$$

Padding behaviour. If the required neighbour count for species b exceeds the number of available neighbours, DeepMD pads the remaining rows using the *last valid* embedding vector. The updated HALMD implementation reproduces this behaviour exactly.

Derivative propagation. For forces, HALMD must compute

$$\frac{\partial G_{ij,k}}{\partial \hat{s}_{ij}} \qquad \text{and} \qquad \frac{\partial \hat{s}_{ij}}{\partial \mathbf{r}_{ij}},$$

which are combined via the chain rule.

The updated filter-network module evaluates

$$\frac{\partial G_{ij}}{\partial \hat{s}_{ij}}$$

using automatic differentiation inside `complete_result()`, ensuring correct multi-species derivative propagation. The previous implementation could not compute neighbour-specific derivatives because it used only one filter network per species.

Summary. The improved G-matrix implementation now includes:

- distinct filter networks $N_\theta^{(a,b)}$ for every species pair,
- runtime selection of the correct network for each neighbour,
- correct handling of normalized inverse distances $\hat{s}_{ij}$,
- optimized reuse of padded embeddings,
- multi-species derivative propagation for accurate forces.

These modifications bring HALMD into full alignment with the DeepMD-v2 specification and enable accurate simulations of multi-component systems.

3.5 Descriptor Computation

The computation of the atomic descriptor follows the DeepPot-SE (`se_e2_a`) formulation of DeepMD-kit v2, in which the local environment of each atom is encoded through a combination of geometric information (R-matrix) and species-aware embedding functions (G-matrix). The resulting descriptor enters the fitting network that predicts the atomic energy E_i and must therefore reproduce the DeepMD-v2 construction exactly.

In the original work of Cruz [1], the descriptor pipeline was implemented for a *single-species* system with a fixed neighbour capacity $N_c^{(a)}$ for each central species a. All descriptor matrices—the geometric matrix R, embedding matrix G, and quadratic descriptor matrix D—were dimensioned accordingly. However, DeepMD-v2 allocates neighbour slots on a *per-species basis*, as defined by the `sel` field of the model:

$$N_c = \sum_{b \in \mathcal{S}} N_c^{(b)},$$

where $N_c^{(b)}$ is the maximum number of neighbours of species b for a central atom of species a. To accommodate this structure, the present work replaces the single neighbour capacity with the multi-species total capacity

$$N_c^{(\mathrm{total})} = \sum_b N_c^{(b)},$$

and enlarges all descriptor matrices accordingly.

R-matrix construction. After periodic extension and neighbour selection (Section 3.4), the coordinate-computation module returns the *normalized geometric matrix*

$$\hat{R} \in \mathbb{R}^{N_c^{(\mathrm{total})} \times (1+d)},$$

whose rows contain the normalized inverse distance $\hat{s}_j$ and the normalized coordinate components. Unlike the earlier HALMD implementation, the rows of $\hat{R}$ are now grouped into species blocks:

$$\underbrace{N_c^{(1)}}_{\text{species 1}} \quad \underbrace{N_c^{(2)}}_{\text{species 2}} \quad \cdots \quad \underbrace{N_c^{(B)}}_{\text{species B}},$$

matching the DeepMD-v2 descriptor layout.

Species-aware G-matrix. DeepMD-v2 employs *distinct filter networks* $N_\theta^{(a,b)}$ for each central–neighbour species pair (a, b). For each neighbour row j, the neighbour-species block determines the appropriate network, and the embedding vector is computed as

$$G_j = N_\theta^{(a,b(j))}(\hat{s}_j)\,,$$

where $\hat{s}_j = \hat{R}_{j,0}$ is the normalized inverse distance (Section 3.4). This replaces the original single-network evaluation $N_\theta^{(a)}(\hat{s}_j)$ used by Cruz and enables correct modelling of multi-component alloys.

The resulting matrix

$$G \in \mathbb{R}^{N_c^{(\mathrm{total})} \times M}$$

is fully species-aware and follows the DeepMD-v2 specification.

Descriptor assembly. DeepMD constructs the descriptor as the quadratic form

$$D = \frac{1}{(N_c^{(\mathrm{total})})^2} \left(G^{\mathsf{T}} \hat{R}\right) \left(\hat{R}^{\mathsf{T}} G_{\mathrm{trunc}}\right),$$

where G_{trunc} contains the first M columns of G, following the standard `se_e2_a` truncation. The original HALMD implementation used the single-species normalization $(N_c^{(a)})^2$; here it is updated to the correct $(N_c^{(\mathrm{total})})^2$, which is necessary for exact agreement with the TensorFlow inference.

The descriptor matrix D is finally flattened row-major to form the input to the fitting network.

Summary of improvements. Compared to Cruz's single-species implementation, the updated descriptor computation:

- generalizes the neighbour axis to multi-species blocks as defined by the `sel` vector;
- evaluates species-dependent embedding networks $N_\theta^{(a,b)}$ for all central–neighbour species pairs;
- expands the geometric and embedding matrices to the full multi-species capacity $N_c^{(\mathrm{total})}$;
- uses the correct DeepMD-v2 normalization $(N_c^{(\mathrm{total})})^2$;
- employs normalized geometric inputs $\hat{R}$ and normalized inverse distances $\hat{s}_j$ consistent with Section 3.4.

This brings HALMD's descriptor pipeline into full alignment with the DeepMD-v2 specification and enables accurate multi-species inference.

3.6 Potential Energy Calculation

In the Deep Potential (DeeP) formalism, the total potential energy of a system is expressed as a sum of atomic energy contributions,

$$E = \sum_{i=1}^{N} E_i, \tag{5}$$

where each atomic energy E_i is obtained by evaluating a species-dependent fitting neural network acting on the descriptor vector $\mathbf{D}_i$ of atom i. For an atom of species s_i, the mapping is

$$E_i = \mathrm{NN}_{s_i}(\mathbf{D}_i), \tag{6}$$

where NN_{s_i} denotes the fitting network associated with species s_i. HALMD evaluates this network using the trained weights extracted from the `frozen_model.pb` file, including DeepMD-v2 activation functions, residual connections, and timestep scaling.

Baseline Implementation Prior to This Work

The original HALMD implementation by Cruz supported:

- **single-species** DeepMD potentials,
- evaluation of a single fitting network for all atoms,
- complete DeepMD feed-forward inference (activations, skip connections, Jacobian computation),
- accurate reproduction of DeepMD energies for monoatomic systems.

However, it lacked the mechanisms required for DeepMD-v2 multi-species inference:

- no species-dependent fitting networks,
- no species-aware descriptor (i.e. no use of $\hat{R}$ or species-partitioned G),
- no support for the per-species atomic energy offset (`bias_atom_e`),

- no coupling between multi-species descriptors and the correct fitting network.

Consequently, HALMD could not reproduce DeepMD-v2 predictions for alloy systems.

Extensions Introduced in This Thesis

This work generalizes the HALMD potential evaluation pipeline to fully support DeepMD-v2 multi-species potentials.

1. Species-dependent fitting networks. DeepMD-v2 defines one fitting network per atomic species. The new implementation stores a vector of networks,

$$\{\mathrm{NN}_0,\ \mathrm{NN}_1,\ \ldots,\ \mathrm{NN}_{S-1}\},$$

and selects the appropriate one according to the central atom's species,

$$E_i = \mathrm{NN}_{s_i}(\mathbf{D}_i).$$

This required extending the `deepmd` operator so that species information is forwarded to both the descriptor and fitting stages.

2. Inclusion of the atomic energy bias term. DeepMD-v2 models may include a constant per-species energy offset `bias_atom_e`. HALMD now adds this term explicitly,

$$E_i = \mathrm{NN}_{s_i}(\mathbf{D}_i) + b_{s_i}, \tag{7}$$

ensuring numerical agreement with DeepMD-kit.

3. Integration with multi-species descriptors. The descriptor introduced in Section 3.4 makes use of:

- the normalized geometric matrix $\hat{R}$,
- species-partitioned neighbour groups,
- species-pair embedding networks $G^{(a,b)}$,
- per-species normalization tensors $(t_{\mathrm{avg}}, t_{\mathrm{std}})$.

so that the fitting network receives the correct DeepMD-v2 descriptor vector $\mathbf{D}_i$.

4. Reproduction of DeepMD-v2 energies. With all extensions enabled, HALMD reproduces DeepMD-v2 energies with floating-point accuracy:

- per-atom energies match the TensorFlow implementation,
- species offsets are correctly incorporated,
- descriptor and fitting network outputs coincide with DeepMD-kit,
- total energies of multi-component systems agree exactly with reference values.

Final Formulation

The atomic energy computed in HALMD now takes the DeepMD-v2 form

$$E_i = \mathrm{NN}_{s_i}(\mathbf{D}_i) + b_{s_i}, \tag{8}$$

and the total potential energy is

$$E_{\mathrm{tot}} = \sum_{i=1}^{N} E_i. \tag{9}$$

This matches DeepMD-v2 inference for all tested single- and multi-species systems.

4 Force Computation

In the Deep Potential (DeeP) framework, forces are obtained as the negative gradient of the total potential energy with respect to atomic coordinates,

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{r}_i}.$$

Because the energy is produced through a multi-stage mapping involving geometric quantities (R), species-dependent embedding networks (G), and a species-dependent fitting network, the force computation requires evaluating a sequence of nested derivatives.

DeepMD expresses this as a structured chain rule passing through:

$$\mathbf{r} \xrightarrow{\text{geometry}} R \xrightarrow{\text{embedding}} G \xrightarrow{\text{fitting network}} E,$$

so that each force component involves contributions from all atoms appearing in the local environment of the corresponding descriptor.

The remainder of this section describes the derivative computation in HALMD. We begin with an overview of automatic differentiation (AD), which is used to evaluate all neural-network derivatives. We then detail the analytic derivatives of the geometry matrix R, the descriptor D, and finally show how these quantities are combined with the fitting-network derivatives to assemble the total atomic forces.

4.1 Automatic Differentiation (AD)

As outlined in the beginning of this section, the computation of atomic forces in DeepMD requires propagating derivatives through a sequence of mappings involving geometric transformations, embedding networks, descriptor algebra, and finally a species-dependent fitting network. While the geometric parts admit closed-form analytic expressions, the neural networks used in DeepMD-v2 contain nonlinear activations, residual connections, and timestep scalings that make analytical differentiation impractical. For these components HALMD employs *automatic differentiation* (AD).

Automatic differentiation is a computational technique for evaluating derivatives of functions represented as compositions of elementary operations [15, 16]. Unlike symbolic differentiation, AD avoids expression explosion, and unlike finite differences, it does not

introduce truncation error. Instead, derivatives are accumulated by applying the chain rule locally at every primitive operation.

Two modes of AD are central:

- **Forward-mode AD**: propagates derivatives together with the computation, efficient when the number of inputs is small.
- **Reverse-mode AD**: propagates derivatives backward from a scalar output, efficient when the number of outputs is one. This is the mechanism underlying backpropagation [17] and the mode used in modern ML frameworks such as TensorFlow [18] and PyTorch [19].

Reverse-mode AD is particularly well suited to DeepMD, since both the embedding networks $G^{(a,b)}$ and the fitting networks NN_{s_i} have scalar inputs or scalar outputs, making derivative evaluation highly efficient.

HALMD uses Boost.Autodiff [20] to compute all derivatives internal to neural networks:

1. For each neighbour j, the filter network $G^{(a,b)}$ provides both the embedding vector G_{ij} and the derivative $\partial G_{ij}/\partial \hat{s}_{ij}$.
2. For each atom i, the fitting network provides both the predicted atomic energy E_i and the Jacobian $\partial E_i/\partial D_{i,\alpha}$.
3. All remaining derivatives—those involving the geometric mapping $R^*(\mathbf{r})$, the scaled distances $\hat{s}_{ij}$, and the descriptor construction $D(G, R^*)$—are evaluated analytically and implemented explicitly in HALMD.

AD therefore supplies exactly the neural-network parts of the chain rule, while the analytical components handle the geometric and descriptor terms. This hybrid strategy achieves:

- efficient multi-species derivative propagation,
- clean separation between neural and geometric derivatives.

The following subsections detail each analytic component of the force derivation: derivatives of the geometric mapping (R^*), descriptor derivatives, fitting-network derivatives, and finally the assembly of the total force.

4.1.1 Derivatives of the Raw Geometry Matrix R^*

The computation of atomic forces requires the derivative of each row of the raw geometry matrix R^* with respect to the coordinates of the central atom. Since R^* depends on both the pairwise displacement $\mathbf{r}_{ij}$ and its magnitude r_{ij}, its derivative naturally splits into *radial* and *angular* components, which are combined to obtain the full geometric derivative.

Structure of the raw geometry row. Recall from Section 3.4.1 that each neighbour j contributes the row

$$R^*_{ij} = \left[\tilde{s}_{ij},\ \tilde{s}_{ij}\frac{x_{ij}}{r_{ij}},\ \tilde{s}_{ij}\frac{y_{ij}}{r_{ij}},\ \tilde{s}_{ij}\frac{z_{ij}}{r_{ij}}\right],$$

where

$$\tilde{s}_{ij} = s_{ij} f_{\mathrm{sw}}(r_{ij}), \qquad s_{ij} = \frac{1}{r_{ij}}.$$

Relative displacement and unit direction. For the central atom i,

$$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i, \qquad r_{ij} = \|\mathbf{r}_{ij}\|, \qquad \hat{\mathbf{r}}_{ij} = \frac{\mathbf{r}_{ij}}{r_{ij}}.$$

(1) Radial derivative: $\partial r_{ij}/\partial \mathbf{r}_i$ The central-atom derivative of the pair distance is

$$\frac{\partial r_{ij}}{\partial \mathbf{r}_i} = -\,\hat{\mathbf{r}}_{ij}, \tag{10}$$

indicating that increasing $\mathbf{r}_i$ in the direction of $\hat{\mathbf{r}}_{ij}$ decreases the separation.

(2) Radial derivative of the switched inverse distance The switched inverse distance is

$$\tilde{s}_{ij} = \frac{1}{r_{ij}} f_{\mathrm{sw}}(r_{ij}).$$

Differentiating w.r.t. r_{ij} yields

$$\frac{\partial \tilde{s}_{ij}}{\partial r_{ij}} = -\frac{1}{r_{ij}^2} f_{\mathrm{sw}}(r_{ij}) + \frac{1}{r_{ij}} \frac{\mathrm{d}f_{\mathrm{sw}}}{\mathrm{d}r}(r_{ij}). \tag{11}$$

Combining Eq. (11) with Eq. (10):

$$\left(\frac{\partial R^*_{ij,0}}{\partial \mathbf{r}_i}\right)_{\mathrm{rad}} = -\,\frac{\partial \tilde{s}_{ij}}{\partial r_{ij}}\,\hat{\mathbf{r}}_{ij}.$$

(3) Radial derivative of angular components For

$$S_\alpha = \tilde{s}_{ij}\frac{d_\alpha}{r_{ij}}, \qquad d_\alpha \in \{x_{ij}, y_{ij}, z_{ij}\},$$

the derivative w.r.t. the distance is

$$\frac{\partial S_\alpha}{\partial r_{ij}} = \frac{d_\alpha}{r_{ij}^2}\frac{\partial \tilde{s}_{ij}}{\partial r_{ij}} - 2\,\tilde{s}_{ij}\frac{d_\alpha}{r_{ij}^3}. \tag{12}$$

Thus,

$$\left(\frac{\partial S_\alpha}{\partial \mathbf{r}_i}\right)_{\mathrm{rad}} = -\,\frac{\partial S_\alpha}{\partial r_{ij}}\hat{\mathbf{r}}_{ij}.$$

(4) Angular derivative: explicit coordinate dependence The components $d_\alpha = x_{j,\alpha} - x_{i,\alpha}$ depend directly on $\mathbf{r}_i$. Holding r_{ij} fixed:

$$\frac{\partial d_\alpha}{\partial \mathbf{r}_i} = -\,\mathbf{e}_\alpha,$$

leading to the angular contribution

$$\left(\frac{\partial S_\alpha}{\partial \mathbf{r}_i}\right)_{\mathrm{ang}} = -\,\frac{\tilde{s}_{ij}}{r_{ij}}\,\mathbf{e}_\alpha. \tag{13}$$

The isotropic term $R^*_{ij,0}$ has no angular component.

(5) Full derivative of the raw geometry row Combining radial and angular parts:

$$\frac{\partial R^*_{ij,\alpha}}{\partial \mathbf{r}_i} = \left(\frac{\partial R^*_{ij,\alpha}}{\partial \mathbf{r}_i}\right)_{\text{rad}} + \left(\frac{\partial R^*_{ij,\alpha}}{\partial \mathbf{r}_i}\right)_{\text{ang}}, \qquad \alpha = 0, 1, 2, 3. \tag{14}$$

(6) Derivative of the normalized geometry row Normalization (Section 3.4.1) gives

$$\hat{R}_{ij,\alpha} = \frac{R^*_{ij,\alpha} - (t_{\text{avg}})_\alpha}{(t_{\text{std}})_\alpha}.$$

Thus,

$$\frac{\partial \hat{R}_{ij,\alpha}}{\partial \mathbf{r}_i} = \frac{1}{(t_{\text{std}})_\alpha} \frac{\partial R^*_{ij,\alpha}}{\partial \mathbf{r}_i}. \tag{15}$$

Summary. The derivative of the geometry matrix R^* includes:

- a **radial component** from the dependence on r_{ij},
- an **angular component** from explicit displacement derivatives,
- a **normalization derivative** for compatibility with DeepMD-v2.

These analytic formulas reproduce the exact geometric derivative pipeline of DeepMD-v2 and provide the geometric contribution needed for force evaluation.

4.2 Descriptor Derivative

4.3 Fitting Network Derivative

4.4 Final Force Assembly

5 Results

5.1 Verification

5.2 Performance Evaluation

5.3 Case Study: Binary Alloy System

6 Discussion

7 Conclusions and Future Work

References

[1] Andres Cruz. “Deep Neural Networks Potentials for Scalable Molecular Dynamics Simulations on Accelerator Hardware”. Master’s Thesis. Freie Universität Berlin, Sept. 2025.

[2] Han Wang, Linfeng Zhang, Jiequn Han, et al. “DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics”. In: *Computer Physics Communications* 228 (2018), pp. 178–184.

[3] Jinzhe Zeng et al. “DeePMD-kit v2: A software package for deep potential models”. In: *The Journal of Chemical Physics* 159.5 (2023).

[4] Peter H Colberg and Felix Höfling. “Highly accelerated simulations of glassy dynamics using GPUs: Caveats on limited floating-point precision”. In: *Computer Physics Communications* 182.5 (2011), pp. 1120–1129.

[5] HALMD Developers. *HALMD: High-Accuracy Large-scale Molecular Dynamics*. https://github.com/halmd-org/halmd. Accessed: 2025-01-10.

[6] Peter H. Colberg and Felix Höfling. *HALMD Documentation and User Manual*. https://halmd.org/doc/. Accessed: 2025-01-10.

[7] Pierre De Buyl et al. “H5MD: A structured, efficient, and portable file format for molecular data”. In: *Computer Physics Communications* 185.6 (2014), pp. 1546–1553. DOI: 10.1016/j.cpc.2014.01.018.

[8] Jörg Behler and Michele Parrinello. “Generalized neural-network representation of high-dimensional potential-energy surfaces”. In: *Physical Review Letters* 98.14 (2007), p. 146401. DOI: 10.1103/PhysRevLett.98.146401.

[9] Frank Noé et al. “Machine learning for molecular simulation”. In: *Annual review of physical chemistry* 71.1 (2020), pp. 361–390.

[10] Albert P Bartók, Risi Kondor, and Gábor Csányi. “On representing chemical environments”. In: *Physical Review B* 87.18 (2013), p. 184115. DOI: 10.1103/PhysRevB.87.184115.

[11] Manzil Zaheer et al. “Deep Sets”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

[12] Albert P Bartók et al. “Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons”. In: *Physical review letters* 104.13 (2010), p. 136403.

[13] Kristof T Schütt et al. “SchNet: A continuous-filter convolutional neural network for modeling quantum interactions”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

[14] Simon Batzner et al. “E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials”. In: *Nature Communications* 13.1 (2022), p. 2453. DOI: 10.1038/s41467-022-29939-5.

[15] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[16] Atilim Gunes Baydin et al. “Automatic differentiation in machine learning: a survey”. In: *Journal of machine learning research* 18.153 (2018), pp. 1–43.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[18] Martín Abadi et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467* (2016).

[19] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[20] “Boost.Autodiff: Automatic Differentiation in Modern C++”. In: *Proceedings of CPP-Now 2023*. 2023.