

Integration of Deep Potential Molecular Dynamics (DeepMD v2) into HALMD for Multi-Species Alloy Systems

Sandip Kumar Sah

M.Sc. Computational Science, Freie Universität Berlin

October 27, 2025

Supervisor: Prof. Dr. Felix Höfling

Outline

- 1 Motivation
- 2 Objectives and Scope
- 3 Background
- 4 Methodology Overview
- 5 Stage 1: Coordinate Normalization and Neighbor List
- 6 Stage 2: Descriptor Formation
- 7 Stage 3: Fitting Network and Energy Evaluation
- 8 Results and Discussion
- 9 Conclusion

- Accurate interatomic potentials are crucial for Molecular Dynamics (MD).
- Classical MD uses empirical potentials (LJ, EAM, etc.) – efficient but approximate.
- **Ab initio MD (AIMD)** provides accuracy but is computationally expensive.
- **Deep Neural Network (DNN) potentials** such as DeepMD bridge this gap:
 - Learn potential energy surface (PES) from DFT data.
 - Achieve near-AIMD accuracy at MD speed.
- Goal: Integrate DeepMD v2 potential inference into **HALMD** for efficient GPU-based alloy simulations.

Research Objectives

- Integrate DeepMD v2 potential into HALMD's many-body potential module.
- Reproduce energy and force predictions directly from extracted model parameters (`frozen_model.pb`).
- Support **multi-species alloy systems** with per-type embedding and fitting networks.
- Maintain computational efficiency suitable for GPU-accelerated MD.

- **HALMD (Highly Accelerated Large-scale Molecular Dynamics):**
 - Modular C++/CUDA architecture.
 - Built-in GPU parallelism and neighbor list handling.
 - Flexible interface for two-body and many-body potentials.
- Provides ideal infrastructure to embed neural-network potentials.

Deep Potential Molecular Dynamics (DeepMD v2)

- DeepMD decomposes total energy into atomic contributions:

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

- Each E_i is predicted by a DNN mapping atomic environments \mathcal{R}_i to energy.
- Two main components:
 - 1 **Filter (Embedding) Network** – encodes local atomic environment.
 - 2 **Fitting Network** – maps descriptors to atomic energies.
- Supports **multi-type systems** with distinct networks per (center, neighbor) pair.

Overview of Implementation

- Workflow between HALMD and DeepMD:
 - ① Extract model weights and statistics from `frozen_model.pb`.
 - ② Reproduce inference pipeline in C++.
 - ③ Compute per-atom energies and forces in HALMD simulation loop.

Normalization and Neighbor List Creation

- Wrap atomic coordinates into the primary periodic cell:

$$\mathbf{r}'_i = \mathbf{H}(\mathbf{H}^{-1}\mathbf{r}_i - \lfloor \mathbf{H}^{-1}\mathbf{r}_i \rfloor)$$

- Create neighbor lists within cutoff radius r_c .
- Ensures consistent local environments under periodic boundaries.
- Step is essential before descriptor computation.

Environment Matrix and Descriptor Construction

- Compute environment matrix \mathbf{R} for each atom:

$$\mathbf{R}_{ij} = s(r_{ij}) \begin{bmatrix} 1/r_{ij} & r_{ij,x}/r_{ij}^2 & r_{ij,y}/r_{ij}^2 & r_{ij,z}/r_{ij}^2 \end{bmatrix}$$

- Apply smooth cutoff $s(r)$ to ensure differentiability.
- Normalize \mathbf{R} using species-specific mean and std.
- Pass scalar component $\hat{s}_{ij} = 1/r_{ij}$ through **filter networks**.

Descriptor Matrix Formation

- Embedding output $\mathbf{G} \in \mathbb{R}^{N_{\text{neigh}} \times M_1}$.
- Descriptor in HALMD (explicit formulation):

$$\mathbf{D}^{(i)} = \frac{1}{N_{\text{neigh}}^2} \mathbf{G}^{\top} (\mathbf{R} \mathbf{R}^{\top}) \mathbf{G}_{<}$$

- Flattened vector $\mathcal{D}_i = \text{vec}(\mathbf{D}^{(i)})$ is used for energy prediction.
- HALMD performs these operations with `boost::ublas` for clarity and derivative consistency.

Fitting Network Structure

- Each atom type t_c has its own fitting MLP.
- Hidden layers:

$$\mathbf{x}^{(l+1)} = \begin{cases} \mathbf{x}^{(l)} + \text{idt}^{(l)} \odot \tanh(\mathbf{x}^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}), & \text{if residual present,} \\ \tanh(\mathbf{x}^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}), & \text{otherwise.} \end{cases}$$

- Final output (linear layer):

$$E_i = \mathbf{x}^{(L)}\mathbf{W}^{(\text{final})} + \mathbf{b}^{(\text{final})} + b_{t_c(i)}$$

- Total potential energy:

$$E_{\text{total}} = \sum_i E_i$$

Implementation Outcomes

- Full inference pipeline implemented in HALMD (C++).
- Supports multiple atomic species (e.g., Cu–Ag alloys).
- Accurately reproduces DeepMD energies and forces.
- Maintains computational efficiency with GPU parallelism.

- Successfully integrated DeepMD v2 inference into HALMD.
- Achieved energy and force consistency across Python and C++ implementations.
- Extended HALMD to handle alloy systems with multi-type neural potentials.
- Lays foundation for future work:
 - Gradient-based force computation.
 - Fully GPU-optimized neural inference.
 - Integration with HALMD's thermostat and ensemble modules.

Thank You!