Accurate implementation of DeepMD-v2 potential calculation in HALMD for single species, extension to multi-species and Automatic-Differentiation-Based Force Computation

Sandip Sah

M.Sc. Computational Science, Freie Universität Berlin

January 07, 2026

Supervisor: Prof. Dr. Felix Höfling

Content I

Motivation I

- Classical potentials are efficient but inaccurate , while *ab initio* methods are accurate but computationally expensive [6, 7].
- Machine-learned potentials (e.g., DeepMD) achieve near–DFT accuracy at classical MD cost [1, 9].
- Current DeepMD–HALMD integration is inaccurate and limited to single-species systems [5].

Objectives and Scope

- Extend the DeepMD integration in HALMD beyond the single-species implementation of Cruz to full *multi-species DeepMD-v2* inference [5].
- Implement the missing core components of the DeepMD-v2 pipeline:
  - periodic coordinate handling, ghost atoms, and multi-type neighbor lists,
  - species-dependent filter and descriptor networks,
  - normalization, scaling, smooth cutoffs, and force calculations.

  [9, 11].
- Focus on the **inference stage** (energy and force evaluation) for multi-component materials using the DeepPot-SE descriptor;
- Model training and alternative descriptors are outside the scope.

HALMD: High-Accuracy Large-Scale MD Engine

- HALMD is a high-performance, open-source MD framework for large-scale simulations, optimized for numerical accuracy and GPU acceleration [4].
- Atomic motion follows Newton's equations of motion:

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

- Traditionally uses efficient interatomic potentials, but are limited for complex chemistry [6, 2].
- This thesis extends HALMD with multi-species DeepMD-v2 force fields, enabling near *ab initio* accuracy at classical MD cost [9, 11].

Neural Network Potentials and Deep Potential MD(DeepMD)-v2

- **Neural Network Potentials** learn interatomic interactions from *ab initio* data, achieving near–DFT accuracy at classical MD cost [1, 8].
- Energy is written as a sum of atomic contributions:

$$E = \sum_i E_i(\mathcal{R}_i), \tag{1}$$

  ensuring locality and linear scaling [1].
- **DeepMD-v2**: modern NNP with smooth, symmetry-preserving descriptors, explicit multi-species support, and efficient GPU execution [9, 11].
- Species-specific embedding and fitting networks predict energies; forces follow from

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{r}_i}. \tag{2}$$

Overview of Implementation

- DeepMD-v2 is integrated into HALMD through a modular, end-to-end inference pipeline.
- Model parameters are extracted from `frozen_model.pb` and `input.json` and converted into a compact HDF5 format for efficient loading.
- HALMD reconstructs local atomic environments using the exact DeepMD-v2 convention:
  - periodic coordinate wrapping,
  - ghost-cell extension,
  - species-ordered neighbor selection.
- From the reconstructed environment, HALMD computes descriptors and evaluates species-specific neural networks to obtain atomic energies and forces.

DeepMD–HALMD Energy Evaluation Pipeline

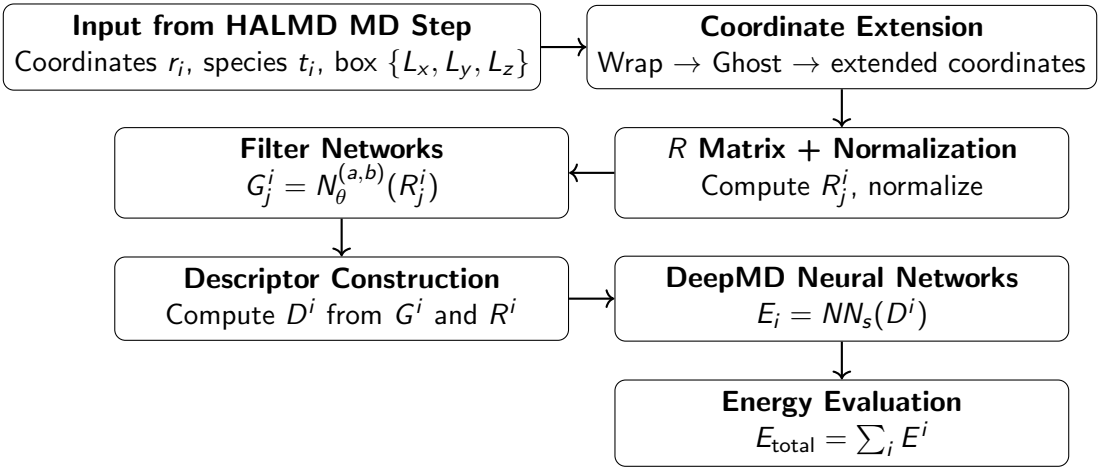


Figure: DeepMD–HALMD energy evaluation pipeline implemented in this thesis.

Model Parameter Extraction

- DeepMD-v2 models are stored as `frozen_model.pb` with hyperparameters in `input.json` [9, 11].
- All parameters are extracted from the TensorFlow graph and converted into a compact HDF5 format for efficient inference in HALMD.
- The extracted model contains:
  - descriptor attributes (cutoffs, normalization, species mapping),
  - species-dependent embedding (filter) networks,
  - species-dependent fitting networks,
  - final outputs: energies, forces, and virials.
- The hierarchical HDF5 model enables full multi-species DeepMD-v2 inference in HALMD, without TensorFlow, extending earlier single-species implementations [5].

Coordinate System Extension

- **Problem:** HALMD's native neighbor handling uses the *minimum-image convention*, sufficient for classical MD, but incompatible with DeepMD's descriptor construction [4, 9, 11].
- **DeepMD Requirement:** Descriptors are computed from *absolute atomic coordinates* inside the primary cell, requiring explicit:
  - coordinate wrapping,
  - ghost-cell periodic expansion,
  - species-aware neighbor organization,
  - descriptor normalization.
- **Implemented Solution:** A new DeePMD-style environment constructor that exactly reproduces the DeepMD-v2 preprocessing pipeline, replacing HALMD's original environment construction [5].
- **Impact:** Enables correct descriptor evaluation and full compatibility with multi-species DeepMD-v2 models.

Explicit Coordinate Wrapping

- DeepMD requires all atoms to be mapped into the primary cell using fractional coordinates:

$$\mathbf{s}_i = \mathbf{r}_i \mathbf{H}^{-1}, \qquad \tilde{\mathbf{s}}_i = \mathbf{s}_i - \lfloor \mathbf{s}_i \rfloor, \qquad \tilde{\mathbf{r}}_i = \tilde{\mathbf{s}}_i \mathbf{H}. \tag{3}$$

- This procedure is mandatory for general (non-orthorhombic) simulation cells and matches DeepMD's internal pipeline [9].

Explicit Coordinate Wrapping visualization

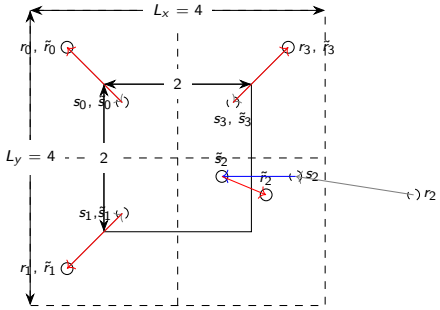


Figure: Explicit coordinate wrapping illustrated in stages: real coordinates r_i, fractional coordinates $s_i = \mathbf{r}_i \mathbf{H}^{-1}$, and wrapped fractional coordinates $\tilde{s}_i$.

Ghost-Cell Periodic Extension

- **Problem:** HALMD's neighbor list applies the minimum-image convention, which is sufficient for classical MD but does *not* provide the full periodic environment required by DeepMD descriptors [4, 9, 11].
- **DeepMD Requirement:** Every atom must retain a complete neighborhood within the cutoff radius r_c, including neighbors across periodic boundaries.
- **Implemented Solution:** Full **ghost-cell tiling** — explicit replication of the simulation cell in all spatial directions before neighbor construction.
- **Impact:** Ensures descriptor correctness and identical environments for atoms near boundaries and in the interior, enabling full DeepMD-v2 compatibility.

Ghost-Cell Construction

- Periodic images are generated from wrapped coordinates:

$$\mathbf{r}_i^{(\mathrm{s})} = \tilde{\mathbf{r}}_i + s_x L_x\,\mathbf{e}_x + s_y L_y\,\mathbf{e}_y + s_z L_z\,\mathbf{e}_z, \quad s_\alpha \in [-n_{\mathrm{buff},\alpha},\, n_{\mathrm{buff},\alpha}] \tag{4}$$

- Buffer size ensures full cutoff coverage:

$$n_{\mathrm{buff},\alpha} = \left\lceil \frac{r_c}{L_\alpha} \right\rceil$$

- For cubic cells with $n_{\mathrm{buff}} = 1$: $3^3 = 27$ replicated cells.

Ghost-Cell construction visualization

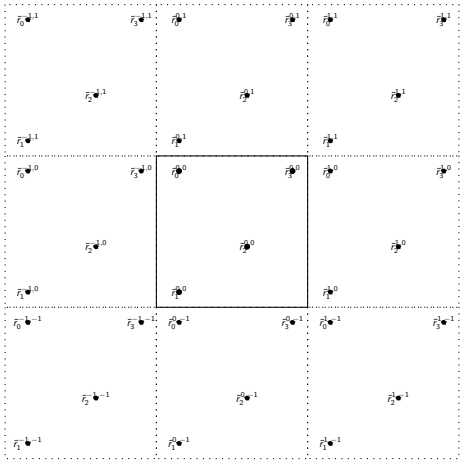


Figure: Periodic replication of the computational cell.

Construction of the Configuration Matrix R I

- **Inputs:** wrapped central coordinate $\tilde{\mathbf{r}}_i$ and ghost-cell neighbour images $\mathbf{r}_j^{(\mathbf{s})}$.
- **Relative displacement:**

$$\mathbf{r}_{ij} = \mathbf{r}_j^{(\mathbf{s})} - \tilde{\mathbf{r}}_i, \qquad r_{ij} = \|\mathbf{r}_{ij}\|. \tag{5}$$

- **Smooth cutoff (DeepMD):**

$$s(r) = \begin{cases} \frac{1}{r}, & r < r_s, \\ \frac{1}{r}\Big[x^3(-6x^2 + 15x - 10) + 1\Big], & r_s \le r < r_c, \\ 0, & r \ge r_c, \end{cases} \qquad x = \frac{r - r_s}{r_c - r_s}. \tag{6}$$

- **Neighbour feature row (4 components):**

$$(R^i)_j = \Big[s(r_{ij}),\ s(r_{ij})\frac{x_{ij}}{r_{ij}},\ s(r_{ij})\frac{y_{ij}}{r_{ij}},\ s(r_{ij})\frac{z_{ij}}{r_{ij}}\Big]. \tag{7}$$

Construction of the Configuration Matrix R II

- **Species-fixed layout via `sel`:**

$$N_c^{(b)} = \text{max neighbour slots for species } b, \qquad \texttt{sel} = [N_c^{(1)}, \ldots, N_c^{(B)}], \tag{8}$$

$$N_c^{(\text{total})} = \sum_{b=1}^{B} N_c^{(b)}, \qquad R^i \in \mathbb{R}^{N_c^{(\text{total})} \times 4}. \tag{9}$$

  Neighbours are grouped by species and truncated/zero-padded to match `sel`.

- **Normalization (training statistics):**

$$\hat{R}^i_{k\alpha} = \frac{R^i_{k\alpha} - t_{\text{avg},\alpha}}{t_{\text{std},\alpha}}, \qquad \hat{s}_{ij} = \hat{R}^i_{j,0} \ \text{(input to embedding net)}. \tag{10}$$

Construction of the Embedding Matrix G I

- The DeepPot-SE descriptor maps each normalized inverse distance

$$\hat{s}_{ij} = \hat{R}^i_{j,0} \tag{11}$$

to a learned embedding vector using a species-pair–specific filter network.

- For each central–neighbor species pair (a, b), DeepMD-v2 defines a dedicated neural network

$$N_\theta^{(a,b)} : \mathbb{R} \to \mathbb{R}^M, \tag{12}$$

producing the embedding row

$$G^i_{j:} = N_\theta^{(a,b(j))}(\hat{s}_{ij}). \tag{13}$$

- All embedding rows are stacked to form

$$G^i \in \mathbb{R}^{N_c^{(\text{total})} \times M}, \quad N_c^{(\text{total})} = \sum_b N_c^{(b)}. \tag{14}$$

Construction of the Embedding Matrix G II

- Different neighbor species use different filter networks, enabling chemically accurate multi-species modeling.
- HALMD stores the full network family as

$$\texttt{networks}[a][b] \equiv N_\theta^{(a,b)}, \tag{15}$$

and selects the correct one at runtime for each neighbor.

Descriptor Computation I

- The DeepPot-SE descriptor combines geometric information $\hat{R}^i$ and learned embeddings G^i to form the final descriptor D^i used to predict atomic energy.
- After neighbor construction, the normalized geometric matrix is

$$\hat{R}^i \in \mathbb{R}^{N_c^{(\text{total})} \times 4}, \tag{16}$$

  containing inverse distances and angular components.
- Each row is embedded via a species-pair neural network:

$$G^i_{j:} = N_\theta^{(a,b(j))}(\hat{s}_{ij}), \qquad G^i \in \mathbb{R}^{N_c^{(\text{total})} \times M}. \tag{17}$$

- The quadratic descriptor is then constructed as

$$D^i = \frac{1}{(N_c^{(\text{total})})^2} \big((G^i)^T \hat{R}^i\big)\big((\hat{R}^i)^T G^{i,\text{trunc}}\big). \tag{18}$$

Descriptor Computation II

- Flattening D^i produces the final descriptor vector

$$\mathbf{D}^i \in \mathbb{R}^{MM'}, \tag{19}$$

which is passed to the fitting network to predict energy and forces.

Potential Energy Evaluation I

- DeepMD expresses the total energy as a sum of atomic energies:

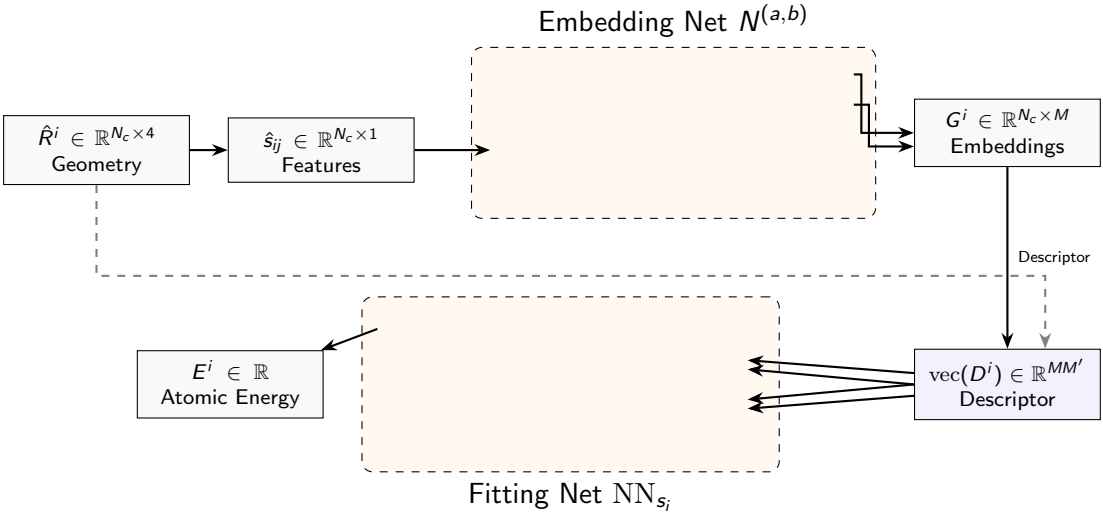
$$E_{\text{tot}} = \sum_{i=1}^{N} E^i. \tag{20}$$

- For each atom i of species s_i, the atomic energy is obtained from a **species-dependent fitting network**:

$$E^i = \text{NN}_{s_i}(\mathbf{D}^i) + b_{s_i}. \tag{21}$$

- DeepMD-v2 defines one fitting network per species; HALMD reconstructs all networks directly from the frozen TensorFlow graph.
- The descriptor vector $\mathbf{D}^i$ is guaranteed to match the exact structure expected by each fitting network (ordering, dimension, normalization).
- The new HALMD implementation extends previous work from single-species to full **multi-species** DeepMD-v2 inference.

summary of potential computation



Energy Agreement

- HALMD reproduces DeepMD energies to single-precision floating-point accuracy.
- Verified for:
  - Cu (Model A) [3],
  - HEA [10],
  - Garnet [10],
  - Cu (Model B with different statistics) [10].
- Agreement holds across:
  - increasing system sizes,
  - chemically complex multi-species systems,
  - different normalization statistics.
- Confirms correctness of:
  - descriptor construction,
  - embedding networks,
  - fitting networks,
  - energy bias terms.

Energy Agreement Across All Models and Species

Model	Species	Atoms	DeepMD (eV)	HALMD (eV)	Match
Cu A	1	4	−6673.84	−6673.84	✓
Cu A	1	32	−53412.4	−53412.4	✓
Cu A	1	108	−180303	−180303	✓
Cu A	1	256	−427425	−427425	✓
Cu B	1	4	−7.7443	−7.7443	✓
Cu B	1	32	−89.4412	−89.4412	✓
Cu B	1	108	−335.96	−335.96	✓
Cu B	1	256	−836.414	−836.414	✓
HEA	5	16	−66.7551	−66.7551	✓
HEA	5	128	−895.498	−895.498	✓
HEA	5	432	−3490.86	−3490.86	✓
Garnet	5	161	−108445	−108445	✓

- Agreement holds for monoatomic and multi-species systems.
- Verified across different training statistics and descriptor normalizations.
- Confirms correctness of full DeepMD-v2 inference in HALMD.

Overview

- In DeepMD, atomic forces are obtained from the energy by

$$\mathbf{F}_i = -\frac{\partial E_{\text{tot}}}{\partial \mathbf{r}_i}. \tag{22}$$

- The total energy depends on atomic coordinates through multiple stages:

$$\mathbf{r} \;\to\; \hat{R} \;\to\; D \;\to\; E.$$

- Force evaluation therefore requires a structured application of the chain rule:

$$\frac{\partial E}{\partial \mathbf{r}} = \frac{\partial E}{\partial D}\frac{\partial D}{\partial \hat{R}}\frac{\partial \hat{R}}{\partial \mathbf{r}}. \tag{23}$$

- HALMD reproduces the DeepMD-v2 derivative pipeline by combining:
  - automatic differentiation for all neural networks,
  - analytic derivatives for geometric and descriptor operations.
- This yields forces that match DeepMD-v2 outputs to floating-point precision.

Automatic Differentiation (AD): How It Works

- **Automatic Differentiation (AD)** computes exact derivatives of a program by applying the chain rule to each elementary operation.
- Key insight: any computation can be decomposed into simple operations

$$+,\ \times,\ \sin,\ \cos,\ \ldots$$

and the chain rule is applied locally to each operation.

- Example computation:

$$z = x\,y + \sin(x)$$

is evaluated as a sequence of primitives:

$$a = xy, \quad b = \sin(x), \quad z = a + b.$$

- AD differentiates this program automatically, without symbolic formulas and without numerical approximations.

AD: Forward-Mode

- **Forward-mode AD**
  - Propagates derivatives together with values.
  - Each variable carries its derivative:

$$(x, \dot{x}),\ (y, \dot{y}),\ (a, \dot{a}), \ldots$$

  - One program run gives the derivative w.r.t. one input.
  - Cost scales with number of inputs.

Forward-Mode AD: Worked Example I

Goal: Compute the derivative of

$$z = x\,y + \sin(x)$$

with respect to x.

Step 1: Decompose computation into primitives

$$a = x\,y$$
$$b = \sin(x)$$
$$z = a + b$$

Step 2: Attach derivatives to every variable

$$(x, \dot{x}),\ (y, \dot{y}),\ (a, \dot{a}),\ (b, \dot{b}),\ (z, \dot{z})$$

Forward-Mode AD: Worked Example II

Seed the input derivative:

$$\dot{x} = 1, \qquad \dot{y} = 0.$$

Step 3: Propagate derivatives using chain rule

$$\begin{aligned} a &= xy, & \dot{a} &= y\,\dot{x} + x\,\dot{y} = y \\ b &= \sin(x), & \dot{b} &= \cos(x)\,\dot{x} = \cos(x) \\ z &= a + b, & \dot{z} &= \dot{a} + \dot{b} = y + \cos(x) \end{aligned}$$

Result:

$$\boxed{\frac{\partial z}{\partial x} = y + \cos(x)}$$

AD: Reverse-Mode

- First run computes all intermediate values.
- Second run propagates sensitivities backward.
- One run yields the full gradient of a scalar output.
- Cost scales with number of outputs.

This makes reverse-mode AD ideal for

$$E = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n,$$

which is exactly the structure of neural networks.

Automatic Differentiation (AD) in HALMD

- DeepMD force evaluation requires derivatives through: geometry $\rightarrow$ embedding $\rightarrow$ descriptor $\rightarrow$ fitting network.
- **Analytic derivatives** are used for all geometric operations.
- **Automatic differentiation** is used for all neural networks.
- **Current HALMD implementation: Forward-mode AD**
  - Used for embedding networks:

$$\hat{s}_{ij} \rightarrow G_j^i, \qquad \frac{\partial G_j^i}{\partial \hat{s}_{ij}}.$$

  - Used in fitting network to compute

$$\frac{\partial E_i}{\partial D_{i,\alpha}}.$$

- HALMD uses a **hybrid strategy**: analytic geometry + AD for neural networks.

Derivatives of the Geometry Matrix R

- Forces require derivatives of each geometry row

$$\frac{\partial R_{j\alpha}^{(i)}}{\partial \mathbf{r}_i}. \tag{24}$$

- Each row depends on the relative displacement

$$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i, \quad r_{ij} = \|\mathbf{r}_{ij}\|. \tag{25}$$

- Geometry features:

$$R_{j0}^{(i)} = s(r_{ij}), \qquad R_{j\alpha}^{(i)} = s(r_{ij})\frac{d_\alpha}{r_{ij}}. \tag{26}$$

- All derivatives follow directly from the multivariate chain rule.

Derivative of the Embedding Matrix G

- Each embedding row is produced by a species-dependent network:

$$G_{j:}^{(i)} = N_{\theta}^{(a,b)}(\hat{s}_{ij}), \qquad \hat{s}_{ij} = \hat{R}_{j0}^{(i)}. \tag{27}$$

- Therefore, $G^{(i)}$ depends on geometry only through the single scalar $\hat{s}_{ij}$.
- Its derivative reduces to a one–dimensional neural-network derivative:

$$\frac{\partial G_{jp}^{(i)}}{\partial \hat{R}_{j0}^{(i)}} = \frac{d\, N_{\theta,p}^{(a,b)}}{d\hat{s}}(\hat{s}_{ij}), \qquad p = 1, \ldots, M. \tag{28}$$

- All other geometric components do **not** affect G:

$$\frac{\partial G_{jp}^{(i)}}{\partial \hat{R}_{j\alpha}^{(i)}} = 0, \qquad \alpha \neq 0. \tag{29}$$

Descriptor Structure and Construction

- For each atom i, the descriptor is built from:

$$\hat{R}^{(i)} \in \mathbb{R}^{N_c \times 4}, \qquad G^{(i)} \in \mathbb{R}^{N_c \times M}.$$

- **Intermediate contraction:**

$$C^{(i)} = \frac{1}{N_c} \left(G^{(i)}\right)^{\mathsf{T}} \hat{R}^{(i)}, \qquad C^{(i)} \in \mathbb{R}^{M \times 4}.$$

- **Quadratic, permutation-invariant descriptor:**

$$D_{\text{ext}}^{(i)} = C^{(i)} \left(C^{(i)}\right)^{\mathsf{T}} \in \mathbb{R}^{M \times M}.$$

- **Truncated descriptor used by the fitting network:**

$$D^{(i)} = D_{\text{ext}}^{(i)}[:, 1 : M'] \;=\; C^{(i)} \left(C^{(i)}\right)^{\mathsf{T}} P \in \mathbb{R}^{M \times M'}.$$

Descriptor Derivative: Key Structure

- Descriptor depends on geometry via two paths:

$$\hat{R}^i \to C^i \to D^i, \qquad \hat{R}^i \to G^i \to C^i \to D^i.$$

- Matrix chain rule:

$$\frac{\partial \operatorname{vec}(D^i)}{\partial \operatorname{vec}(\hat{R}^i)} = \underbrace{\frac{\partial \operatorname{vec}(D^i)}{\partial \operatorname{vec}(G^i)} \frac{\partial \operatorname{vec}(G^i)}{\partial s^i} \frac{\partial s^i}{\partial \operatorname{vec}(\hat{R}^i)}}_{\text{embedding path}} + \underbrace{\frac{\partial \operatorname{vec}(D^i)}{\partial \operatorname{vec}(\hat{R}^i)}}_{\text{direct geometry path}} .$$

Fitting Network Derivative

- Atomic energy:

$$E^i = \mathrm{NN}_{s_i}\big(\mathrm{vec}(D^i)\big)$$

- Jacobian required for forces:

$$J^i = \frac{\partial E^i}{\partial D^i} = \frac{\partial E^i}{\partial\, \mathrm{vec}(D^i)} \in \mathbb{R}^{M \times M'}.$$

- Deep nonlinear mapping $\rightarrow$ **automatic differentiation (AD)**.

Final Force Assembly

- Total force:

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

- Final factored force expression:

$$\mathbf{F}_i = -\sum_j J^j \left(\underbrace{\frac{\partial \operatorname{vec}(D_j)}{\partial \operatorname{vec}(G_j)} \frac{\partial \operatorname{vec}(G_j)}{\partial s_j} \frac{\partial s_j}{\partial \operatorname{vec}(\hat{R}_j)}}_{\text{via embedding } (\hat{R} \to s \to G \to D)} + \underbrace{\frac{\partial \operatorname{vec}(D_j)}{\partial \operatorname{vec}(\hat{R}_j)}}_{\text{direct geometry } (\hat{R} \to D)} \right) \frac{\partial \operatorname{vec}(\hat{R}_j)}{\partial \mathbf{r}_i}$$

Force Computation: Hybrid Structure

- **Neural part:**

$$\frac{\partial \operatorname{vec}(D)}{\partial \operatorname{vec}(G)}, \quad \frac{\partial \operatorname{vec}(G)}{\partial s} \quad \text{(via AD)}$$

- **Analytic geometry:**

$$\frac{\partial s}{\partial \operatorname{vec}(\hat{R})}, \quad \frac{\partial \operatorname{vec}(\hat{R})}{\partial \mathbf{r}}$$

- Fully explicit, stable and DeepMD-v2–equivalent force pipeline.

Conclusion & Outlook

- First fully explicit *DeepMD-v2 implementation in HALMD* with multi-species descriptors, energies, and force derivatives.
- Verified energy agreement for monoatomic and complex multi-species systems enables *accurate large-scale ML-driven molecular dynamics* without framework overhead.
- This work enables:
  - faster ML-based simulations,
  - improved HPC scalability,
  - closer integration of ML and physics-based modeling.
- With completed forces and further optimisation, HALMD can become a *high-performance platform for next-generation materials simulation*.

References I

[1] 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.

[2] Donald W. Brenner et al. "A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons". In: *Journal of Physics: Condensed Matter* 14.4 (2002), pp. 783–802. DOI: 10.1088/0953-8984/14/4/312.

[3] Matteo Cioni, Daniela Polino, and Daniele Rapetti. *Cu_fcc_slabs Deep Potential Model*. www.aissquare.com/models/detail?pageType=models&name=Cu_fcc_slabs. Machine-learned interatomic potential trained with DeePMD-kit. 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.

References II

[5] Andres Cruz. "Deep Neural Networks Potentials for Scalable Molecular Dynamics Simulations on Accelerator Hardware". Master's Thesis. Freie Universität Berlin, Sept. 2025.

[6] Murray S. Daw and Michael I. Baskes. "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals". In: *Physical Review B* 29.12 (1984), pp. 6443–6453. DOI: 10.1103/PhysRevB.29.6443.

[7] Dominik Marx and Jürg Hutter. *Ab initio molecular dynamics: basic theory and advanced methods*. Cambridge University Press, 2009.

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

[9] 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.

References III

[10] Jinzhe Zeng and Duo Zhang. *Data for DeePMD-kit v2 paper*. https://github.com/deepmodeling-activity/deepmd-kit-v2-paper. Supporting data for: DeePMD-kit v2: A software package for deep potential models, J. Chem. Phys., 159, 054801 (2023). 2023.

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

Thank You!