Supplementary Materials for *Deep Potential Molecular Dynamics: a scalable model with the accuracy of quantum mechanics*

Linfeng Zhang and Jiequn Han
Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544, USA

Han Wang*
Institute of Applied Physics and Computational Mathematics, Fenghao East Road 2, Beijing 100094, P.R. China and CAEP Software Center for High Performance Numerical Simulation, Huayuan Road 6, Beijing 100088, P.R. China

Roberto Car
Department of Chemistry, Department of Physics, Program in Applied and Computational Mathematics, Princeton Institute for the Science and Technology of Materials, Princeton University, Princeton, NJ 08544, USA

Weinan E†
Department of Mathematics and Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544, USA and Center for Data Science, Beijing International Center for Mathematical Research, Beijing Institute of Big Data Research, Beijing, 100871, P.R. China

Keywords:

TRAINING/TESTING DATA

water and ice

The data used for training and/or testing are extracted from the AIMD simulations summarized in Tab. I. All the simulations adopt a time step of 0.48 fs. The PI-AIMD simulations use the CPMD codes of Quantum Espresso [9] for the DFT part and are interfaced with the i-PI code [2] for the path-integral part. The generalized Langevin equation with color noise [1] in i-PI requires 8 beads for a converged representation of the Feynman paths. The classical AIMD simulations use the CPMD codes of Quantum Espresso and adopt the Nosé-Hoover thermostat [7] for thermalization. The Parrinello-Rahman technique [8] for variable cell dynamics is adopted in all cases. The training datasets include 95000 snapshots (from 105000 total snapshots) randomly selected along the liquid water trajectory, 19500 snapshots (from 24000 total snapshots) randomly selected along the ice (b) trajectory, 9500 snapshots (from 12000 total snapshots) randomly selected along the ice (c) trajectory, and 9500 snapshots (from 12000 total snapshots) randomly selected along the ice (d) trajectory. The remaining snapshots in the database are used for testing purposes.

molecules

The data and their complete description for the organic molecules (benzene, uracil, napthalene, aspirin, salicylic acid, malonaldehyde, ethanol, and toluene) can be found at http://quantum-machine.org/. For each molecule, 95000 snapshots, randomly selected from the database, are used to train the DeePMD model. The remaining snapshots in

TABLE I: Equilibrated AIMD trajectories (traj.) for liquid water (LW) and ice Ih.

System	PI/classical	N	P [bar]	T [K]	traj. length [ps]
LW	path integral	64	1.0	300	6.2
ice (b)	path integral	96	1.0	273	1.5
ice (c)	classical	96	1.0	330	6.0
ice (d)	classical	96	2.13k	238	6.0

the database are used for testing purposes.

IMPLEMENTATION OF THE METHOD

The TensorFlow r1.0 software library (http://tensorflow.org/) is interfaced with our C++ codes for data training and for calculating the energy, the forces, and the virial.

network input data

We consider a system consisting of N atoms. The global coordinates of the atoms, in the laboratory frame, are $\{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_N\}$, where $\boldsymbol{R}_i = \{x_i, y_i, z_i\}$ for each i. The neighbors of atom i are denoted by $\mathcal{N}(i) = \{j : |\boldsymbol{R}_{ij}| < R_c\}$, where $\boldsymbol{R}_{ij} = \boldsymbol{R}_i - \boldsymbol{R}_j$, and R_c is the cut-off radius. The neighbor list $\mathcal{N}(i)$ is sorted according to the scheme illustrated in Fig. 1. In extended systems, the number of neighbors at different snapshots inside R_c fluctuates. Let N_c be the largest fluctuating number of neighbors. The two atoms used to define the axes of the local frame of atom i are called the axis-atoms and are denoted by $a(i) \in \mathcal{N}(i)$ and $b(i) \in \mathcal{N}(i)$, respectively. In general we choose two closest atoms, independently of their species, together with the center atom, to define the local frame. Thus, in all the water cases, we choose the other two atoms belonging to the same water molecule. We apply the same rule to the organic molecules, but in this case we exclude the hydrogen atoms in the definition of the axis-atoms.

Next, we define the rotation matrix $\mathcal{R}(\boldsymbol{R}_{ia(i)}, \boldsymbol{R}_{ib(i)})$ for the local frame of atom i,

$$\mathcal{R}(\boldsymbol{R}_{ia(i)}, \boldsymbol{R}_{ib(i)}) = \begin{pmatrix} \boldsymbol{e}[\boldsymbol{R}_{ia(i)}] \\ \boldsymbol{e}[\boldsymbol{R}_{ib(i)} - (\boldsymbol{R}_{ia(i)} \cdot \boldsymbol{R}_{ib(i)} \boldsymbol{R}_{ia(i)}] \\ \boldsymbol{e}[\boldsymbol{R}_{ia(i)} \times \boldsymbol{R}_{ib(i)}] \end{pmatrix}^T , \tag{1}$$

where $\boldsymbol{e}[\boldsymbol{x}] \equiv \frac{\boldsymbol{x}}{||\boldsymbol{x}||}$. In this local frame of reference, we obtain the new set of coordinates:

$$\boldsymbol{R}'_{ij} = \left\{x'_{ij}, y'_{ij}, z'_{ij}\right\} = \{x_{ij}, y_{ij}, z_{ij}\}\, \mathcal{R}(\boldsymbol{R}_{ia(i)}, \boldsymbol{R}_{ib(i)}), \tag{2}$$

and we define $R'_{ij} = ||\boldsymbol{R}'_{ij}||$. Then the spacial information for $j \in \mathcal{N}(i)$ is

$$\boldsymbol{D}_{ij} \equiv \begin{cases} \left\{D^0_{ij}, D^1_{ij}, D^2_{ij}, D^3_{ij}\right\} = \left\{\frac{1}{R'_{ij}}, \frac{x'_{ij}}{R'^2_{ij}}, \frac{y'_{ij}}{R'^2_{ij}}, \frac{z'_{ij}}{R'^2_{ij}}\right\}, & \text{full radial and angular information;} \\ \left\{D^0_{ij}\right\} = \left\{\frac{1}{R'_{ij}}\right\}, & \text{radial information only.} \end{cases}$$

When $\alpha = 0, 1, 2, 3$, full (radial plus angular) information is provided. When $\alpha = 0$, only radial information is used. Note that for $j \in \mathcal{N}(i)$, D^α_{ij} is a function of the global coordinates of three or four atoms:

$$D^\alpha_{ij} = \begin{cases} D^\alpha_{ij}(\boldsymbol{R}_i, \boldsymbol{R}_{a(i)}, \boldsymbol{R}_{b(i)}), & \text{for } j = a(i) \text{ or } j = b(i); \\ D^\alpha_{ij}(\boldsymbol{R}_i, \boldsymbol{R}_{a(i)}, \boldsymbol{R}_{b(i)}, \boldsymbol{R}_j), & \text{otherwise.} \end{cases}$$

This formula is useful in the derivation of the formulae for the forces and the virial tensor given below.

The neural network uses a fixed input data size. Thus, when the size of $\mathcal{N}(i)$ is smaller than N_c, we temporarily set to zero the input nodes not used for storing the D^α_{ij}. The nodes set to zero are still labeled by D^α_{ij}.

The D^α_{ij} are then standardized to be the input data for the neural networks. In this procedure, the D^α_{ij} are grouped according to the different atomic species. Within each group we calculate the mean and standard deviation of each D^α_{ij} by averaging over the snapshots of the training sample and over all the atoms in the group. Then we shift the D^α_{ij} by their corresponding means, and divide them by their corresponding standard deviations. Because of the weight $1/R$ in the D^α_{ij} and because the unoccupied nodes are set to zero, some standard deviations are very small or even zero. This causes an ill-posed training process. Therefore, after the shift operations, we divide by 0.01 Å^{-1} the D^α_{ij} with standard deviation smaller than 0.01 Å^{-1}. For simplicity, we still use the same notation for the standardized D^α_{ij}.

deep neural network for the energy

For atom i, the "atomic energy" is represented as

$$E_i = N_{\boldsymbol{w}(i)}(\{D^{\alpha}_{ij}\}_{j\in\mathcal{N}(i),\alpha}), \tag{3}$$

where $N_{\boldsymbol{w}(i)}$ is the network that computes the atomic contribution to the total energy, and $\boldsymbol{w}(i)$ are the weights used to parametrize the network, which depend on the chemical species of atom i.

In this work, $N_{\boldsymbol{w}(i)}$ is constructed as a feedforward network in which data flows from the input layer as $\{D^{\alpha}_{ij}\}$, through multiple fully connected hidden layers, to the output layer as the atomic energy E_i. More specifically, a feedforward neural network with N_h hidden layers is a mapping

$$\mathcal{N}_i(\{D^{\alpha}_{ij}\}) = \mathcal{L}^{\text{out}}_i \circ \mathcal{L}^{N_h}_i \circ \mathcal{L}^{N_h-1}_i \circ \cdots \circ \mathcal{L}^{1}_i(\{D^{\alpha}_{ij}\}), \tag{4}$$

where the symbol "$\circ$" denotes function composition. Here $\mathcal{L}^p_i$ is the mapping from layer $p-1$ to p, a composition of a linear transformation and a non-linear transformation

$$\boldsymbol{d}^p_i = \mathcal{L}^p_i(\boldsymbol{d}^{p-1}_i) = \varphi\big(\boldsymbol{W}^p_i \boldsymbol{d}^{p-1}_i + \boldsymbol{b}^p_i\big). \tag{5}$$

The $\boldsymbol{d}^p_i \in \mathbb{R}^{M_p}$ denote the values of neurons in layer p and M_p the number of neurons. The weight matrix $\boldsymbol{W}^p_i \in \mathbb{R}^{M_p\times M_{p-1}}$ and bias vector $\boldsymbol{b}^p_i \in \mathbb{R}^{M_p}$ are free parameters of the linear transformation that are to be optimized. The non-linear activation function φ is in general a component-wise function, and here it is taken to be the hyperbolic tangent, i.e.,

$$\varphi(d_1, d_2, \ldots, d_M) = (\tanh(d_1), \tanh(d_2), \ldots, \tanh(d_M)). \tag{6}$$

The output mapping $\mathcal{L}^{\text{out}}_i$ is a linear transformation,

$$\mathcal{L}^{\text{out}}_i(\boldsymbol{d}^{N_h}_i) = \boldsymbol{W}^{\text{out}}_i \boldsymbol{d}^{N_h-1} + b^{\text{out}}_i, \tag{7}$$

with weight vector $\boldsymbol{W}^{\text{out}}_i \in \mathbb{R}^{1\times M_{N_h}}$ and bias $b^{\text{out}}_i \in \mathbb{R}$ being free parameters to be optimized as well. On the whole, all the free parameters associated with atom i are

$$\boldsymbol{w}(i) = \{\boldsymbol{W}^1_i, \boldsymbol{b}^1_i, \boldsymbol{W}^2_i, \boldsymbol{b}^2_i, \cdots, \boldsymbol{W}^{N_h}_i, \boldsymbol{b}^{N_h}_i, \boldsymbol{W}^{\text{out}}_i, b^{\text{out}}_i\}. \tag{8}$$

It should be stressed that, to guarantee the permutational symmetry, atoms of the same species share the same parameters $\boldsymbol{w}$.

forces and virial tensor

The total potential energy is the sum of the E_i. Thus the forces are

$$\begin{aligned}
\boldsymbol{F}_i = -\nabla_{\boldsymbol{R}_i} E &= -\sum_j \nabla_{\boldsymbol{R}_i} E_j = -\sum_j \sum_{k\in\mathcal{N}(j)} \nabla_{\boldsymbol{R}_i} N_{\boldsymbol{w}(j)}(\{D^{\alpha}_{jk}\}_{k\in\mathcal{N}(j),\alpha}) \\
&= -\sum_j \sum_{k\in\mathcal{N}(j)} \sum_{\alpha} \frac{\partial N_{\boldsymbol{w}(j)}}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} \\
&= -\sum_{k\in\mathcal{N}(i)} \sum_{\alpha} \frac{\partial N_{\boldsymbol{w}(i)}}{\partial D^{\alpha}_{ik}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{ik} - \sum_{j\neq i} \sum_{k\in\mathcal{N}(j)} \sum_{\alpha} \delta(i-a(j)) \frac{\partial N_{\boldsymbol{w}(j)}}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} \\
&\quad - \sum_{j\neq i} \sum_{k\in\mathcal{N}(j)} \sum_{\alpha} \delta(i-b(j)) \frac{\partial N_{\boldsymbol{w}(j)}}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} - \sum_{j\neq i} \sum_{k\in\tilde{\mathcal{N}}(j)} \sum_{\alpha} \delta(i-k) \frac{\partial N_{\boldsymbol{w}(j)}}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} \\
&= -\sum_{k\in\mathcal{N}(i)} \sum_{\alpha} \frac{\partial E_i}{\partial D^{\alpha}_{ik}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} - \sum_{j\neq i} \sum_{k\in\mathcal{N}(j)} \sum_{\alpha} \delta(i-a(j)) \frac{\partial E_j}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} \\
&\quad - \sum_{j\neq i} \sum_{k\in\mathcal{N}(j)} \sum_{\alpha} \delta(i-b(j)) \frac{\partial E_j}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk} - \sum_{j\neq i} \sum_{k\in\tilde{\mathcal{N}}(j)} \sum_{\alpha} \delta(i-k) \frac{\partial E_j}{\partial D^{\alpha}_{jk}} \nabla_{\boldsymbol{R}_i} D^{\alpha}_{jk},
\end{aligned}$$

where $\tilde{\mathcal{N}}(j) = \mathcal{N}(j)\backslash\{a(j), b(j)\}$.

The virial tensor is defined as $\Xi_{\alpha\beta} = -\frac{1}{2}\sum_i R_{i\alpha}F_{i\beta}$, where the indices α and β indicate Cartesian components in the lab reference frame. Due to the periodic boundary conditions, one cannot directly use the absolute coordinates $R_{i\alpha}$ to compute the virial tensor. Rather, in the AIMD framework, the virial tensor is defined with an alternative but equivalent formula, i.e.,

$$\Xi_{\alpha\beta} = -\frac{1}{2}\sum_{\gamma}\frac{\partial E}{\partial h_{\alpha\gamma}}h_{\gamma\beta}, \tag{9}$$

where h is the cell tensor. In our framework, due to the decomposition of the local energy E_i, one computes the virial tensor by:

$$\Xi_{\alpha\beta} = -\frac{1}{2}\sum_{i,j} x_{\alpha}^{(i,j)} f_{\beta}^{(i,j)}. \tag{10}$$

$x_{\alpha}^{(i,j)}$ is the α-th component of the vector oriented from the i-th to the j-th atom in the difference:

$$x_{\alpha}^{(i,j)} = x_{\alpha}^{(i)} - x_{\alpha}^{(j)}. \tag{11}$$

$f_{\beta}^{(i,j)}$ is the β-th component of the negative gradient of E_i w.r.t. x_j, i.e.,

$$f_{\beta}^{(i,j)} = -\frac{\partial E_i}{\partial_{x_j^{\beta}}}. \tag{12}$$

Together with the energy representation described above, all the quantities needed for training and MD simulations, although complicated, have been analytically defined. In particular, it is noted that the derivatives of the total energy with respect to the atomic positions, appearing in both the forces and the viral tensor, are computed by the chain rule through the backpropagation algorithm, provided by TensorFlow. To make it work, we additionally implement the computation of $\nabla_{\boldsymbol{R}_i} D_{jk}^{\alpha}$ in C++ and interface it with TensorFlow.

Training Details

During the training process, one minimizes the family of loss functions defined in the paper:

$$L(p_\epsilon, p_f, p_\xi) = p_\epsilon \Delta\epsilon^2 + \frac{p_f}{3N}\sum_i |\Delta \boldsymbol{F}_i|^2 + \frac{p_\xi}{9}||\Delta\xi||^2. \tag{13}$$

The network weights are optimized with the Adam stochastic gradient descent method [5]. An initial learning rate $r_{l0} = 0.001$ is used with the Adam parameters set to β_1=0.9, β_2=0.999, and ϵ=1.0$\times$10^{-8}, which are the default settings in TensorFlow. The learning rate r_l decays exponentially with the global step:

$$r_l = r_{l0} d_r^{-c_s/d_s}, \tag{14}$$

where d_r, c_s, and d_s are the decay rate, the global step, and the decay step, respectively. In this paper, the batch size is 4 in all the training processes. The decay rate is 0.95. For liquid water, the training process undergoes 4000000 steps in total, and the learning rate is updated every 20000 steps. For molecules, the training process undergoes 8000000 steps in total, and the learning rate is updated every 40000 steps.

We remark that, for the prefactors, a proper linear evolution with the learning rate speeds up dramatically the training process. We define this process by:

$$p = p_{limit}(1 - \frac{r_l}{r_{l0}}) + p_{start}(\frac{r_l}{r_{l0}}), \tag{15}$$

in which p_{start} is the prefactor at the beginning of the training process, and p_{limit} is approximately the prefactor at the end. We define p_{start} for the energy, the forces, and the virial as p_{estart}, p_{fstart}, and p_{vstart}, respectively.

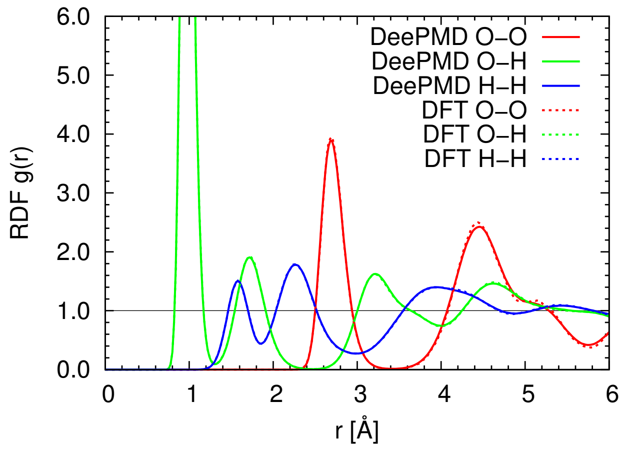

FIG. S1: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (b).

Similarly, we define p_{limit} for the energy, the forces, and the virial as p_{elimit}, p_{flimit}, and p_{vlimit}, respectively. In this paper, we use the following scheme:

$$\begin{cases} p_{estart} = 1, & p_{elimit} = 400; \\ p_{fstart} = 1000, & p_{flimit} = 1, \end{cases} \tag{16}$$

for both water and the molecules, and

$$\begin{cases} p_{vstart} = 1, p_{vlimit} = 400, & \text{for liquid water and ice (b);} \\ p_{vstart} = 0, p_{vlimit} = 0, & \text{for ice (c) and (d) and the molecules.} \end{cases} \tag{17}$$

The above scheme is based on the following considerations. Each snapshot of the AIMD trajectories provides 1 energy, $3N$ forces, and 6 independent virial tensor elements. The number of force components is much larger than the number of energy and virial tensor components. Therefore, matching the forces at the very beginning of the training process makes the training efficient. As the training proceeds, increasing the prefactors of the energy and the virial tensor allows us to achieve a well balanced training in which the energy, the forces, and the virial are mutually consistent.

In the original Deep Potential paper [3], only the energy was used to train the network, requiring in some cases the use of Batch Normalization techniques [4] to deal with issues of overfitting and training efficiency. Adding the forces and/or the virial tensor provides a strong regularization of the network and makes training significantly more efficient. Thus Batch Normalization techniques are not necessary within the DeePMD framework.

DeePMD details

In the path-integral/classical NPT simulations of liquid water and ice, we integrate our codes with the i-PI software. The DeePMD simulations are performed at the same thermodynamic conditions, and use the same temperature and pressure controls, of the corresponding AIMD simulations. All DeePMD trajectories for water and ice are 300 ps long and use the same time step of the AIMD simulations.

We use our own code to perform the constant temperature MD simulations of the organic molecules. In each DeePMD simulation the temperature is the same of that of the corresponding AIMD simulation. The time step and time length of the trajectories in these simulations are the same of those in the corresponding AIMD trajectories.

ADDITIONAL RESULTS

The radial distribution functions (RDFs) of ice Ih (b), (c) ,and (d) are reported in Figs. S1, S2, and S3, respectively.

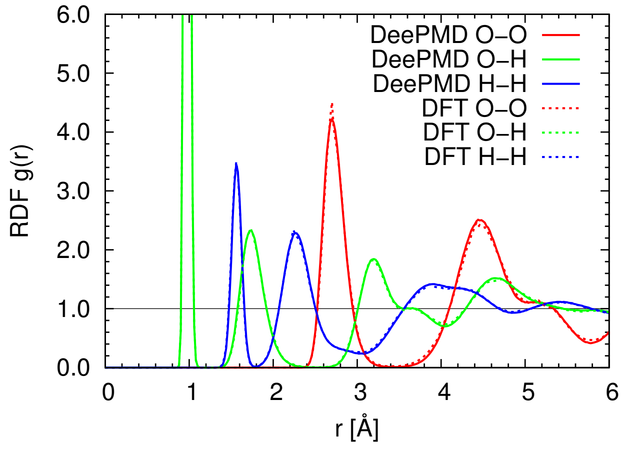

FIG. S2: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (c).

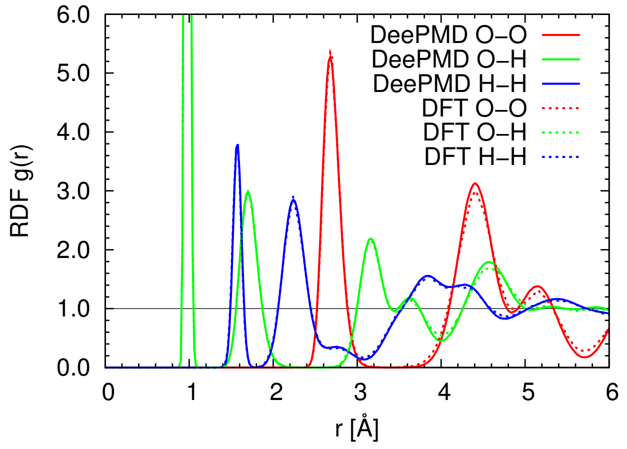

FIG. S3: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (d).

The probability distribution function of the O-O bond orientation order parameter Q_6 is reported in Fig. S4. The bond orientation order parameter for oxygen i, as proposed in Ref. [6], is defined by

$$Q_l(i) = \left[\frac{4\pi}{2l+1}\sum_{m=-l}^{l}|\bar{q}_{lm}(i)|^2\right]^{\frac{1}{2}}, \tag{18}$$

where

$$\bar{q}_{lm}(i) = \frac{\sum_{j\in\tilde{N}_b(i)} s(r_{ij})q_{lm}(j)}{\sum_{j\in\tilde{N}_b(i)} s(r_{ij})}, \quad q_{lm}(i) = \frac{\sum_{j\in N_b(i)} s(r_{ij})Y_{lm}(\hat{\boldsymbol{r}}_{ij})}{\sum_{j\in N_b(i)} s(r_{ij})}, \tag{19}$$

and $\tilde{N}_b(i) = N_b(i) \cup \{i\}$. The $Y_{lm}(\cdots)$ denotes the spherical harmonic function, the $N_b(i)$ denotes the set of oxygen neighbors of oxygen i, and the $s(r_{ij})$ is a switching function defined by

$$s(r) = \begin{cases} 1, & r < r_{min}; \\ \frac{1}{2} + \frac{1}{2}\cos\left(\pi\frac{r - r_{min}}{r_{max} - r_{min}}\right), & r_{min} \le r < r_{max}; \\ 0, & r \ge r_{max}. \end{cases} \tag{20}$$

In this work we take $r_{min} = 0.31$ nm and $r_{max} = 0.36$ nm.

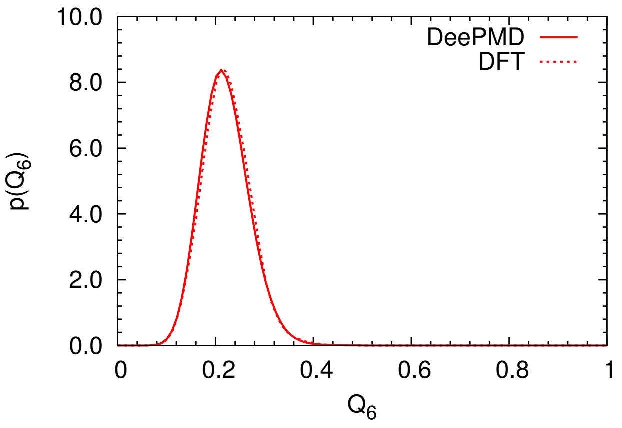

FIG. S4: Probability distribution function of the O-O bond orientation order parameter Q_6

[*] Electronic address: wang˙han@iapcm.ac.cn

[†] Electronic address: weinan@math.princeton.edu

[1] Ceriotti, M., Manolopoulos, D. E., and Parrinello, M., The Journal of Chemical Physics **134**, 084104 (2011).

[2] Ceriotti, M., More, J., and Manolopoulos, D. E., Computer Physics Communications **185**, 1019 (2014).

[3] Han, J., Zhang, L., Car, R., and E, W., arXiv Preprint arXiv:1707.01478 (2017).

[4] Ioffe, S. and Szegedy, C., in *Proceedings of The 32nd International Conference on Machine Learning (ICML)* (2015).

[5] Kingma, D. and Ba, J., in *Proceedings of the International Conference on Learning Representations (ICLR)* (2015).

[6] Lechner, W. and Dellago, C., The Journal of chemical physics **129**, 114707 (2008).

[7] Martyna, G. J., Klein, M. L., and Tuckerman, M., The Journal of Chemical Physics **97**, 2635 (1992).

[8] Parrinello, M. and Rahman, A., Physical Review Letters **45**, 1196 (1980).

[9] http://www.quantum-espresso.org/