

# 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<sup>†</sup>

*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,  
Peking University, Beijing Institute of Big Data Research, Beijing, 100871, P.R. China*

## Abstract

We introduce a scheme for molecular simulations, the Deep Potential Molecular Dynamics (DeePMD) method, based on a many-body potential and interatomic forces generated by a carefully crafted deep neural network trained with *ab initio* data. The neural network model preserves all the natural symmetries in the problem. It is “first principle-based” in the sense that there are no *ad hoc* components aside from the network model. We show that the proposed scheme provides an efficient and accurate protocol in a variety of systems, including bulk materials and molecules. In all these cases, DeePMD gives results that are essentially indistinguishable from the original data, at a cost that scales linearly with system size.

---

\*Electronic address: wang'han@iapcm.ac.cn

†Electronic address: weinan@math.princeton.edu

Molecular dynamics (MD) is used in many disciplines, including physics, chemistry, biology, and materials science, but its accuracy depends on the model for the atomic interactions. *Ab initio* molecular dynamics (AIMD) [1, 2] has the accuracy of density functional theory (DFT) [3], but its computational cost limits typical applications to hundreds of atoms and time scales of  $\sim 100$  ps. Applications requiring larger cells and longer simulations are currently accessible only with empirical force fields (FFs) [4–6], but the accuracy and transferability of these models is often in question.

Developing FFs is challenging due to the many-body character of the potential energy. Expansions in 2- and 3-body interactions may capture the physics [7] but are strictly valid only for weakly interacting systems. A large class of potentials, including the embedded atom method (EAM) [8], the bond order potentials [9], and the reactive FFs [10], share the physically motivated idea that the strength of a bond depends on the local environment, but the functional form of this dependence can only be given with crude approximations.

Machine learning (ML) methodologies are changing this state of affairs [11–20]. When trained on large datasets of atomic configurations and corresponding potential energies and forces, ML models can reproduce the original data accurately. In training these models, the atomic coordinates cannot be used as they appear in MD trajectories because their format does not preserve the translational, rotational, and permutational symmetry of the system. Different ML models address this issue in different ways. Two successful schemes are the Behler-Parrinello neural network (BPNN) [13] and the gradient-domain machine learning (GDML) [19]. In BPNN symmetry is preserved by mapping the coordinates onto a large set of two- and three-body symmetry functions, which are, however, largely *ad hoc*. Fixing the symmetry functions may become painstaking in systems with many atomic species. In GDML the same goal is achieved by mapping the coordinates onto the eigenvalues of the Coulomb matrix, whose elements are the inverse distances between all distinct pairs of atoms. It is not straightforward how to use the Coulomb matrix in extended periodic systems. So far GDML has only been used for relatively small molecules.

In this letter we introduce an NN scheme for MD simulations, called Deep Potential Molecular Dynamics (DeePMD), which overcomes the limitations associated to auxiliary quantities like the symmetry functions or the Coulomb matrix. In our scheme a local reference frame and a local environment is assigned to each atom. Each environment contains a finite number of atoms, whose local coordinates are arranged in a symmetry preserving way

following the prescription of the Deep Potential method [3], an approach that was devised to train an NN with the potential energy only. With typical AIMD datasets this is insufficient to reproduce the trajectories. DeePMD overcomes this limitation. In addition, the learning process in DeePMD improves significantly over the Deep Potential method thanks to the introduction of a flexible family of loss functions. The NN potential constructed in this way reproduces accurately the AIMD trajectories, both classical and quantum (path integral), in extended and finite systems, at a cost that scales linearly with system size and is always several orders of magnitude lower than that of equivalent AIMD simulations.

In DeePMD the potential energy of each atomic configuration is a sum of “atomic energies”,  $E = \sum_i E_i$ , where  $E_i$  is determined by the local environment of atom  $i$  within a cutoff radius  $R_c$  and can be seen as a realization of the embedded atom concept. The environmental dependence of  $E_i$ , which embodies the many-body character of the interactions, is complex and nonlinear. The NN is able to capture the analytical dependence of  $E_i$  on the coordinates of the atoms in the environment in terms of the composition of the sequence of mappings associated to the individual hidden layers. The additive form of  $E$  naturally preserves the extensive character of the potential energy. Due to the analyticity of the “atomic energies” DeePMD is, in principle, a conservative model.

$E_i$  is constructed in two steps. First, a local coordinate frame is set up for every atom and its neighbors inside  $R_c$  [35]. This allows us to preserve the translational, rotational, and permutational symmetries of the environment, as shown in Fig. 1, which illustrates the format adopted for the local coordinate information ( $\{\mathbf{D}_{ij}\}$ ). The  $1/R_{ij}$  factor present in  $\mathbf{D}_{ij}$  reduces the weight of the particles that are more distant from atom  $i$ .

Next,  $\{\mathbf{D}_{ij}\}$  serves as input of a deep neural network (DNN) [22], which returns  $E_i$  in output (Fig. 2). The DNN is a feed forward network, in which data flow from the input layer to the output layer ( $E_i$ ), through multiple hidden layers consisting of several nodes that input the data  $d_l^{\text{in}}$  from the previous layer and output the data  $d_k^{\text{out}}$  to the next layer. A linear transformation is applied to the input data, i.e.,  $\tilde{d}_k = \sum_l w_{kl} d_l^{\text{in}} + b_k$ , followed by action of a non-linear function  $\varphi$  on  $\tilde{d}_k$ , i.e.,  $d_k^{\text{out}} = \varphi(\tilde{d}_k)$ . In the final step from the last hidden layer to  $E_i$ , only the linear transformation is applied. The composition of the linear and nonlinear transformations introduced above provides the analytical representation of  $E_i$  in terms of the local coordinates. The technical details of this construction are discussed in the supplementary materials (SM). In our applications, we adopt the hyperbolic tangent

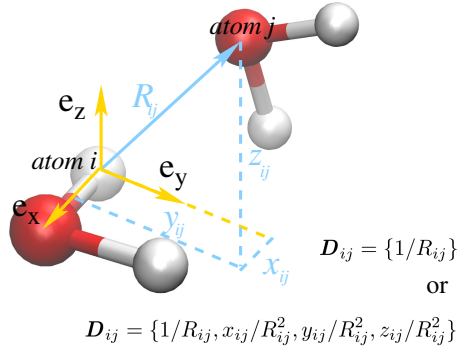


FIG. 1: (color online). Schematic plot of the neural network input for the environment of atom  $i$ , taking water as an example. Atom  $j$  is a generic neighbor of atom  $i$ .  $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$  is the local frame of atom  $i$ .  $\mathbf{e}_x$  is along the O-H bond.  $\mathbf{e}_z$  is perpendicular to the plane of the water molecule.  $\mathbf{e}_y$  is the cross product of  $\mathbf{e}_z$  and  $\mathbf{e}_x$ .  $(x_{ij}, y_{ij}, z_{ij})$  are the Cartesian components of the vector  $\mathbf{R}_{ij}$  in this local frame.  $R_{ij}$  is the length of  $\mathbf{R}_{ij}$ . The neural network input  $\mathbf{D}_{ij}$  may either contain the full radial and angular information of atom  $j$ , i.e.,  $\mathbf{D}_{ij} = \{1/R_{ij}, x_{ij}/R_{ij}^2, y_{ij}/R_{ij}^2, z_{ij}/R_{ij}^2\}$ , or only the radial information, i.e.,  $\mathbf{D}_{ij} = \{1/R_{ij}\}$ . We first sort the neighbors of atom  $i$  according to their chemical species, e.g. oxygens first then hydrogens. Within each species we sort the atoms according to their inverse distances to atom  $i$ , i.e.,  $1/R_{ij}$ . We use  $\{\mathbf{D}_{ij}\}$  to denote the sorted input data for atom  $i$ .

for  $\varphi$  and use 5 hidden layers with decreasing number of nodes per layer, i.e., 240, 120, 60, 30, and 10 nodes, respectively, from the innermost to the outermost layer. It is known empirically that the hidden layers greatly enhance the capability of neural networks to fit complex and highly nonlinear functional dependences [23, 24]. In our case, only by including a few hidden layers could DeePMD reproduce the trajectories with sufficient accuracy.

We use the Adam method [5] to optimize the parameters  $w_{kl}$  and  $b_k$  of each layer with the family of loss functions

$$L(p_\epsilon, p_f, p_\xi) = p_\epsilon \Delta \epsilon^2 + \frac{p_f}{3N} \sum_i |\Delta \mathbf{F}_i|^2 + \frac{p_\xi}{9} \|\Delta \xi\|^2. \quad (1)$$

Here  $\Delta$  denotes the difference between the DeePMD prediction and the training data,  $N$  is the number of atoms,  $\epsilon$  is the energy per atom,  $\mathbf{F}_i$  is the force on atom  $i$ , and  $\xi$  is the virial tensor  $\Xi = -\frac{1}{2} \sum_i \mathbf{R}_i \otimes \mathbf{F}_i$  divided by  $N$ . In Eq. (A13),  $p_\epsilon$ ,  $p_f$ , and  $p_\xi$  are tunable prefactors. When virial information is missing from the data, we set  $p_\xi = 0$ . In order to minimize the loss function in Eq. (A13) in a well balanced way, we vary the magnitude of

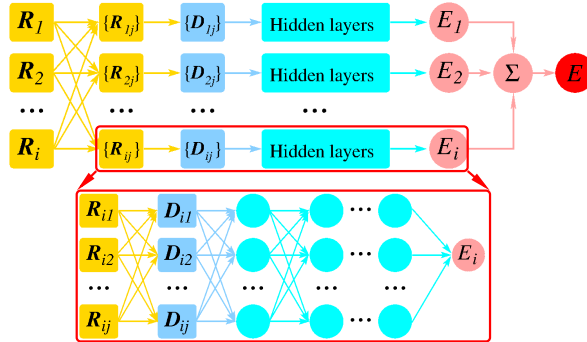


FIG. 2: (color online). Schematic plot of the DeePMD model. The frame in the box is the zoom-in of a DNN. The relative positions of all neighbors w.r.t. atom  $i$ , i.e.,  $\{\mathbf{R}_{ij}\}$ , is first converted to  $\{\mathbf{D}_{ij}\}$ , then passed to the hidden layers to compute  $E_i$ .

the prefactors during training. We progressively increase  $p_\epsilon$  and  $p_\xi$  and decrease  $p_f$ , so that the force term dominates at the beginning, while energy and virial terms become important at the end. We find that this strategy is very effective and reduces the total training time to a few core hours in all the test cases.

To test the method, we have applied DeePMD to extended and finite systems. As representative extended systems, we consider (a) liquid water at  $P = 1$  bar and  $T = 300$  K, at the PI-AIMD level, (b) ice Ih at  $P = 1$  bar and  $T = 273$  K, at the PI-AIMD level, (c) ice Ih at  $P = 1$  bar and  $T = 330$  K, at the classical AIMD level, and (d) ice Ih at  $P = 2.13$  kbar and  $T = 238$  K, which is the experimental triple point for ice I, II, and III, at the classical AIMD level. The variable periodic simulation cell contains 64  $\text{H}_2\text{O}$  molecules in the case of liquid water, and 96  $\text{H}_2\text{O}$  molecules in the case of ices. We adopt  $R_c = 6.0$  Å and use the full radial and angular information for the 16 oxygens and the 32 hydrogens closest to the atom at the origin, while retaining only radial information for all the other atoms within  $R_c$ . All the ice simulations include proton disorder. Deuterons replace protons in the simulations (c) and (d). The PBE0+TS [26, 27] functional is adopted in all cases. As representative finite systems we consider benzene, uracil, naphthalene, aspirin, salicylic acid, malonaldehyde, ethanol, and toluene, for which classical AIMD trajectories with the PBE+TS functional [27, 28] are available [36]. In these systems, we set  $R_c$  large enough to include all the atoms, and use the full radial and angular information in each local frame.

We discuss the performance of DeePMD according to four criteria: (i) generality of the model; (ii) accuracy of the energy, forces, and virial tensor; (iii) faithfulness of the

TABLE I: The RMSE of the DeePMD prediction for water and ices in terms of the energy, the forces, and/or the virial. The RMSEs of the energy and the virial are normalized by the number of molecules in the system.

System	Energy [meV]	Force [meV/Å]	Virial [meV]
liquid water	1.0	40.4	2.0
ice Ih (b)	0.7	43.3	1.5
ice Ih (c)	0.7	26.8	-
Ice Ih (d)	0.8	25.4	-

trajectories; (*iv*) scalability and computational cost. We refer to the SM for full details on the DeePMD implementation and the training datasets.

*Generality.* Bulk and molecular systems exhibit different levels of complexity. The liquid water samples include quantum fluctuations. The organic molecules differ in composition and size, and the corresponding datasets include large numbers of conformations. Yet DeePMD produces satisfactory results in all cases, using the same methodology, network structure, and optimization scheme. The excellent performance of DeePMD in systems so diverse suggests that the method should be applicable to harder systems such as biological molecules, alloys, and liquid mixtures.

*Accuracy.* We quantify the accuracy of energy, forces, and virial predictions in terms of the root mean square error (RMSE) in the case of water and ices (Tab. I), and in terms of the mean absolute error (MAE) in the case of the organic molecules (Tab. II). No virial information was used for the latter. In the water case, the RMSE of the forces is comparable to the accuracy of the minimization procedure in the original AIMD simulations, in which the allowed error in the forces was less than  $10^{-3}$  a.u.. In the case of the molecules, the predicted energy and forces are generally slightly better than the GDML benchmark.

*MD trajectories.* In the case of water and ices, we perform path-integral/classical DeePMD simulations at the thermodynamic conditions of the original models, using the i-PI software [2], but with much longer simulation time (300 ps). The average energy  $\bar{E}$ , density  $\bar{\rho}$ , radial distribution functions (RDFs), and a representative angular distribution function (ADF), i.e., a 3-body correlation function, are reproduced with high accuracy. The results are summarized in Tab. III. The RDFs and ADF of the quantum trajectories of water

TABLE II: The MAE of the DeePMD prediction for organic molecules in terms of the energy and the forces. The numbers in parentheses are the GDML results [19].

Molecule	Energy [meV]	Force [meV/Å]
Benzene	2.8 (3.0)	7.6 (10.0)
Uracil	3.7 (4.0)	9.8 (10.4)
Naphthalene	4.1 (5.2)	7.1 (10.0)
Aspirin	8.7 (11.7)	19.1 (42.9)
Salicylic acid	4.6 (5.2)	10.9 (12.1)
Malonaldehyde	4.0 (6.9)	12.7 (34.7)
Ethanol	2.4 (6.5)	8.3 (34.3)
Toluene	3.7 (5.2)	8.5 (18.6)

TABLE III: The equilibrium energy and density,  $\bar{E}$  and  $\bar{\rho}$ , of water and ices, with DeePMD and AIMD. The numbers in square brackets are the AIMD results. The numbers in parentheses are statistical uncertainties in the last one or two digits. The training AIMD trajectories for the ices are shorter and more correlated than in the water case.

System	$\bar{E}$ [eV/H <sub>2</sub> O]	$\bar{\rho}$ [g/m <sup>3</sup> ]
liquid water	-467.678(2) [-467.679(6)]	1.013(5) [1.013(20)]
ice Ih (b)	-467.750(1) [-467.747(4)]	0.967(1) [0.966(6)]
ice Ih (c)	-468.0478(3) [-468.0557(16)]	0.950(1) [0.949(2)]
ice Ih (d)	-468.0942(2) [-468.1026(9)]	0.986(1) [0.985(2)]

are shown in Fig. 3. The RDFs of ice are reported in the SM. A higher-order correlation function, the probability distribution function of the O-O bond orientation order parameter  $Q_6$ , is additionally reported in the SM and shows excellent agreement between DeePMD and AIMD trajectories. In the case of the molecules, we perform DeePMD at the same temperature of the original data, using a Langevin thermostat with a damping time  $\tau = 0.1$  ps. The corresponding distributions of interatomic distances are very close to the original data (Fig. 4).

*Scalability and computational cost.* All the physical quantities in DeePMD are sums of local contributions. Thus, after training on a relatively small system, DeePMD can be



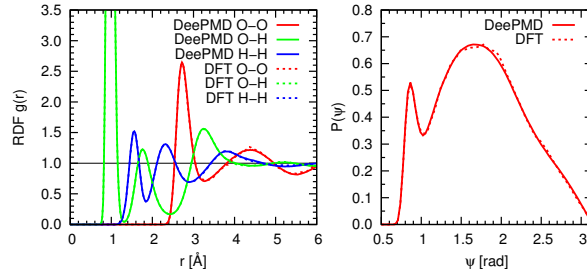


FIG. 3: Correlation functions of liquid water from DeePMD and PI-AIMD. Left: RDFs. Right: the O-O-O ADF within a cutoff radius of 3.7 Å.

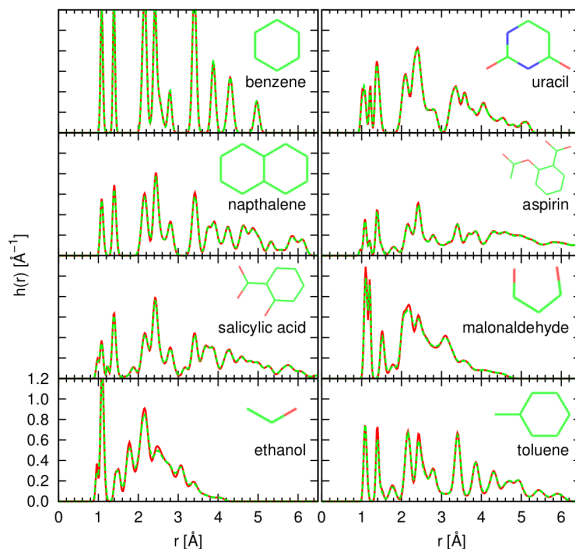


FIG. 4: Interatomic distance distributions of the organic molecules. The solid lines denote the DeePMD results. The dashed lines denote the AIMD results.

directly applied to much larger systems. The computational cost of DeePMD scales linearly with the number of atoms. Moreover, DeePMD can be easily parallelized due to its local decomposition and the near-neighbor dependence of its “atomic energies”. In Fig. 5, we compare the cost of DeePMD fixed-cell simulations ( $NVT$ ) of liquid water with that of equivalent simulations with AIMD and the empirical FF TIP3P [30] in units of CPU core seconds/step/molecule.

While in principle the environmental dependence of  $E_i$  is analytical, in our implementation discontinuities are present in the forces, due to adoption of a sharp cutoff radius, limitation of angular information to a fixed number of atoms, and abrupt changes in the atomic lists due to sorting. These discontinuities are similar in magnitude to those present in the

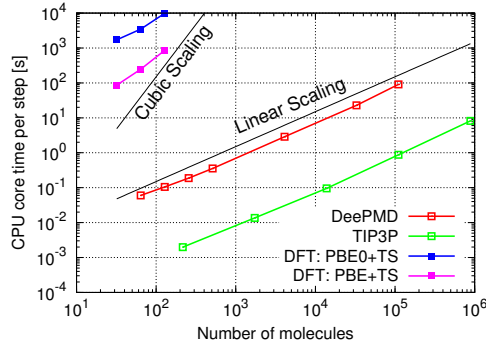


FIG. 5: Computational cost of MD step *vs.* system size, with DeePMD, TIP3P, PBE+TS and, PBE0+TS. All simulations are performed on a Nersc Cori supercomputer with the Intel Xeon CPU E5-2698 v3. The TIP3P simulations use the Gromacs codes (version 4.6.7) [31]. The PBE+TS and PBE0+TS simulations use the Quantum Espresso codes [32].

AIMD forces due to finite numerical accuracy in the enforcement of the Born-Oppenheimer condition. In both cases, the discontinuities are much smaller than thermal fluctuations and perfect canonical evolution is achieved by coupling the systems to a thermostat. We further note that long-range Coulomb interactions are not treated explicitly in the current implementation, although implicitly present in the training data. Explicit treatment of Coulombic effects may be necessary in some applications and deserves further study.

In conclusion, DeePMD realizes a paradigm for molecular simulation, wherein accurate quantum mechanical data are faithfully parametrized by machine learning algorithms, which make possible simulations of DFT based AIMD quality on much larger systems and for much longer time than with direct AIMD. While substantially more predictive than empirical FFs, DFT is not chemically accurate [37]. In principle DeePMD could be trained with chemically accurate data from high-level quantum chemistry [33] and/or quantum Monte Carlo [34], but so far this has been prevented by the large computational cost of these calculations.

DeePMD should also very useful to coarse grain the atomic degrees of freedom, for example, by generating an NN model for a reduced set of degrees of freedom while using the full set of degrees of freedom for training. The above considerations suggest that DeePMD should enhance considerably the realm of AIMD applications by successfully addressing the dilemma of accuracy versus efficiency that has confronted the molecular simulation community for a long time.

## Acknowledgments

The authors acknowledge H.-Y. Ko and B. Santra for sharing the AIMD data on water and ice. The work of J. Han and W. E is supported in part by Major Program of NNSFC under grant 91130005, ONR grant N00014-13-1-0338, DOE grants de-sc0008626 and de-sc0009248, and NSFC grant U1430237. The work of R. Car is supported in part by DOE-SciDAC grant de-sc0008626. The work of H. Wang is supported by the National Science Foundation of China under Grants 11501039 and 91530322, the National Key Research and Development Program of China under Grants 2016YFB0201200 and 2016YFB0201203, and the Science Challenge Project No. JCKY2016212A502.

- 
- [1] R. Car and M. Parrinello, *Physical Review Letters* **55**, 2471 (1985).
  - [2] D. Marx and J. Hutter, *Ab initio molecular dynamics: basic theory and advanced methods* (Cambridge University Press, 2009).
  - [3] W. Kohn and L. J. Sham, *Physical Review* **140**, A1133 (1965).
  - [4] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. and Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov, and A. Mackerell Jr., *Journal of Computational Chemistry* **31**, 671 (2010).
  - [5] W. Jorgensen, D. Maxwell, and J. Tirado-Rives, *Journal of the American Chemical Society* **118**, 11225 (1996).
  - [6] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, *Journal of Computational Chemistry* **25**, 1157 (2004).
  - [7] F. H. Stillinger and T. A. Weber, *Physical Review B* **31**, 5262 (1985).
  - [8] M. S. Daw and M. I. Baskes, *Physical Review B* **29**, 6443 (1984).
  - [9] D. W. Brenner, O. A. Shenderova, J. A. Harrison, S. J. Stuart, B. Ni, and S. B. Sinnott, *Journal of Physics: Condensed Matter* **14**, 783 (2002).
  - [10] A. C. Van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard, *The Journal of Physical Chemistry A* **105**, 9396 (2001).
  - [11] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, *Journal of Computational Physics* **285**, 316 (2015).

- [12] T. D. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen, and R. Ramprasad, *NPJ Computational Materials* **3**, 1 (2017).
- [13] J. Behler and M. Parrinello, *Physical Review Letters* **98**, 146401 (2007).
- [14] J. Behler, *The Journal of Chemical Physics* **145**, 170901 (2016).
- [15] T. Morawietz, A. Singraber, C. Dellago, and J. Behler, *Proceedings of the National Academy of Sciences*, 201602375 (2016).
- [16] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, *Physical Review Letters* **104**, 136403 (2010).
- [17] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. VonLilienfeld, *Physical Review Letters* **108**, 058301 (2012).
- [18] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, *Nature Communications* **8**, 13890 (2017).
- [19] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, *Science Advances* **3**, e1603015 (2017).
- [20] J. S. Smith, O. Isayev, and A. E. Roitberg, *Chemical Science* **8**, 3192 (2017).
- [3] J. Han, L. Zhang, R. Car, and W. E, *Communications in Computational Physics* **23**, 629 (2018).
- [22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT Press, 2016).
- [23] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, in *Advances in neural information processing systems* (2007) pp. 153–160.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in neural information processing systems* (2012) pp. 1097–1105.
- [5] D. Kingma and J. Ba, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2015).
- [26] C. Adamo and V. Barone, *The Journal of Chemical Physics* **110**, 6158 (1999).
- [27] A. Tkatchenko and M. Scheffler, *Physical Review Letters* **102**, 073005 (2009).
- [28] J. P. Perdew, K. Burke, and M. Ernzerhof, *Physical Review Letters* **77**, 3865 (1996).
- [2] M. Ceriotti, J. More, and D. E. Manolopoulos, *Computer Physics Communications* **185**, 1019 (2014).
- [30] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, *The Journal of Chemical Physics* **79**, 926 (1983).

- [31] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. Shirts, J. Smith, P. Kasson, D. van der Spoel, B. Hess, and E. Lindahl, *Bioinformatics*, btt055 (2013).
- [32] O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, *et al.*, *Journal of Physics: Condensed Matter* (2017).
- [33] J. D. Watts, J. Gauss, and R. J. Bartlett, *The Journal of Chemical Physics* **98**, 8718 (1993).
- [34] D. Ceperley and B. Alder, *Science* **231** (1986).
- [35] Some flexibility can be used in the definition of the local frame of atom  $i$ . Usually we define it in terms of the two atoms closest to  $i$ , independently of their species. Exceptions to this rule are discussed in the SM.
- [36] <http://quantum-machine.org/>
- [37] Conventionally, chemical accuracy corresponds to an error of 1 kcal/mol in the energy.

## Appendix A: Supplementary Materials

### 1. Training/Testing Data

#### *a. water and ice*

The data used for training and/or testing are extracted from the AIMD simulations summarized in Tab. IV. 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.

TABLE IV: 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

*b. molecules*

The data and their complete description for the organic molecules (benzene, uracil, naphthalene, 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 the database are used for testing purposes.

**2. 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.

*a. network input data*

We consider a system consisting of  $N$  atoms. The global coordinates of the atoms, in the laboratory frame, are  $\{R_1, R_2, \dots, R_N\}$ , where  $R_i = \{x_i, y_i, z_i\}$  for each  $i$ . The neighbors of atom  $i$  are denoted by  $\mathcal{N}(i) = \{j : |R_{ij}| < R_c\}$ , where  $R_{ij} = R_i - R_j$ , 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  $\mathcal{R}(R_{ia(i)}, R_{ib(i)})$  for the local frame of atom  $i$ ,

$$\mathcal{R}(R_{ia(i)}, R_{ib(i)}) = \begin{pmatrix} e[R_{ia(i)}] \\ e[R_{ib(i)} - (R_{ia(i)} \cdot R_{ib(i)})R_{ia(i)}] \\ e[R_{ia(i)} \times R_{ib(i)}] \end{pmatrix}^T, \quad (\text{A1})$$

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

$$R'_{ij} = \{x'_{ij}, y'_{ij}, z'_{ij}\} = \{x_{ij}, y_{ij}, z_{ij}\} \mathcal{R}(R_{ia(i)}, R_{ib(i)}), \quad (\text{A2})$$

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

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

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

$$D_{ij}^\alpha = \begin{cases} D_{ij}^\alpha(R_i, R_{a(i)}, R_{b(i)}), & \text{for } j = a(i) \text{ or } j = b(i); \\ D_{ij}^\alpha(R_i, R_{a(i)}, R_{b(i)}, R_j), & \text{otherwise.} \end{cases}$$

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  $D_{ij}^\alpha$ . The nodes set to zero are still labeled by  $D_{ij}^\alpha$ .

The  $D_{ij}^\alpha$  are then standardized to be the input data for the neural networks. In this procedure, the  $D_{ij}^\alpha$  are grouped according to the different atomic species. Within each group we calculate the mean and standard deviation of each  $D_{ij}^\alpha$  by averaging over the snapshots of the training sample and over all the atoms in the group. Then we shift the  $D_{ij}^\alpha$  by their corresponding means, and divide them by their corresponding standard deviations. Because of the weight  $1/R$  in the  $D_{ij}^\alpha$  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  $0.01 \text{ \AA}^{-1}$  the  $D_{ij}^\alpha$  with standard deviation smaller than  $0.01 \text{ \AA}^{-1}$ . For simplicity, we still use the same notation for the standardized  $D_{ij}^\alpha$ .

*b. deep neural network for the energy*

For atom  $i$ , the “atomic energy” is represented as

$$E_i = N_{w(i)}(\{D_{ij}^\alpha\}_{j \in \mathcal{N}(i), \alpha}), \quad (\text{A3})$$

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

In this work,  $N_{w(i)}$  is constructed as a feedforward network in which data flows from the input layer as  $\{D_{ij}^\alpha\}$ , 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

$$\mathcal{N}_i(\{D_{ij}^\alpha\}) = \mathcal{L}_i^{\text{out}} \circ \mathcal{L}_i^{N_h} \circ \mathcal{L}_i^{N_h-1} \circ \dots \circ \mathcal{L}_i^1(\{D_{ij}^\alpha\}), \quad (\text{A4})$$

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

$$\mathbf{d}_i^p = \mathcal{L}_i^p(\mathbf{d}_i^{p-1}) = \varphi(\mathbf{W}_i^p \mathbf{d}_i^{p-1} + \mathbf{b}_i^p). \quad (\text{A5})$$

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

$$\varphi(d_1, d_2, \dots, d_M) = (\tanh(d_1), \tanh(d_2), \dots, \tanh(d_M)). \quad (\text{A6})$$

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

$$\mathcal{L}_i^{\text{out}}(\mathbf{d}_i^{N_h}) = \mathbf{W}_i^{\text{out}} \mathbf{d}_i^{N_h} + b_i^{\text{out}}, \quad (\text{A7})$$

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

$$w(i) = \{\mathbf{W}_i^1, \mathbf{b}_i^1, \mathbf{W}_i^2, \mathbf{b}_i^2, \dots, \mathbf{W}_i^{N_h}, \mathbf{b}_i^{N_h}, \mathbf{W}_i^{\text{out}}, b_i^{\text{out}}\}. \quad (\text{A8})$$

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



c. forces and virial tensor

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

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

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

The virial tensor is defined as  $\Xi_{\alpha\beta} = -\frac{1}{2} \sum_i R_{i\alpha} F_{i\beta}$ , where the indices  $\alpha$  and  $\beta$  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.,

$$\Xi_{\alpha\beta} = -\frac{1}{2} \sum_{\gamma} \frac{\partial E}{\partial h_{\alpha\gamma}} h_{\gamma\beta}, \quad (\text{A9})$$

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:

$$\Xi_{\alpha\beta} = -\frac{1}{2} \sum_{i,j} x_{\alpha}^{(i,j)} f_{\beta}^{(i,j)}. \quad (\text{A10})$$

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

$$x_{\alpha}^{(i,j)} = x_{\alpha}^{(i)} - x_{\alpha}^{(j)}. \quad (\text{A11})$$

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

$$f_\beta^{(i,j)} = -\frac{\partial E_i}{\partial x_j^\beta}. \quad (\text{A12})$$

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  $\nabla_{R_i} D_{jk}^\alpha$  in C++ and interface it with TensorFlow.

#### *d. Training Details*

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

$$L(p_\epsilon, p_f, p_\xi) = p_\epsilon \Delta \epsilon^2 + \frac{p_f}{3N} \sum_i |\Delta \mathbf{F}_i|^2 + \frac{p_\xi}{9} \|\Delta \xi\|^2. \quad (\text{A13})$$

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  $\beta_1=0.9$ ,  $\beta_2=0.999$ , and  $\epsilon=1.0 \times 10^{-8}$ , which are the default settings in TensorFlow. The learning rate  $r_l$  decays exponentially with the global step:

$$r_l = r_{l0} d_r^{-c_s/d_s}, \quad (\text{A14})$$

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:

$$p = p_{limit} \left(1 - \frac{r_l}{r_{l0}}\right) + p_{start} \left(\frac{r_l}{r_{l0}}\right), \quad (\text{A15})$$

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. 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:

$$\begin{cases} p_{estart} = 1, & p_{elimit} = 400; \\ p_{fstart} = 1000, & p_{flimit} = 1, \end{cases} \quad (\text{A16})$$

for both water and the molecules, and

$$\begin{cases} p_{vstart} = 1, p_{vlimit} = 400, & \text{for liquid water and ice (b);} \\ p_{vstart} = 0, p_{vlimit} = 0, & \text{for ice (c) and (d) and the molecules.} \end{cases} \quad (\text{A17})$$

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.

#### *e. 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.

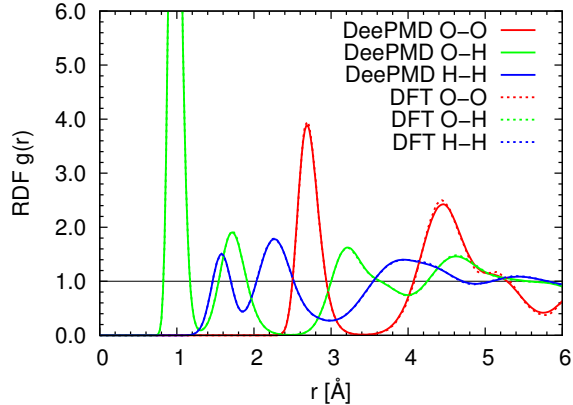


FIG. 6: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (b).

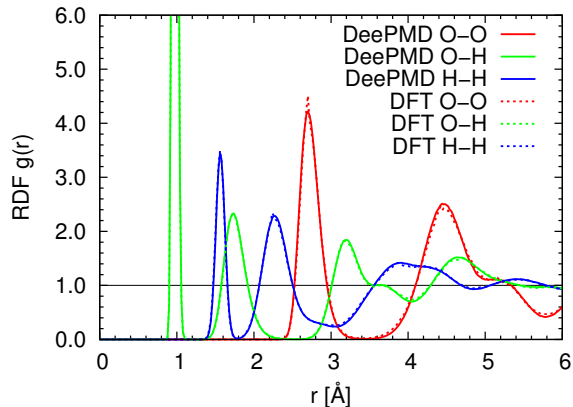


FIG. 7: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (c).

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.

### 3. Additional Results

The radial distribution functions (RDFs) of ice Ih (b), (c), and (d) are reported in Figs. 6, 7, and 8, respectively.

The probability distribution function of the O-O bond orientation order parameter  $Q_6$  is reported in Fig. 9. The bond orientation order parameter for oxygen  $i$ , as proposed in

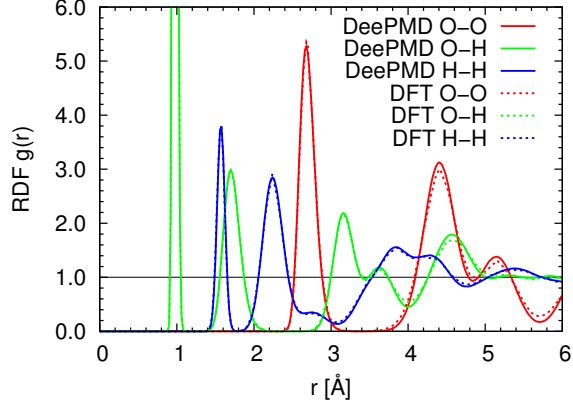


FIG. 8: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (d).

Ref. [6], is defined by

$$Q_l(i) = \left[ \frac{4\pi}{2l+1} \sum_{m=-l}^l |\bar{q}_{lm}(i)|^2 \right]^{\frac{1}{2}}, \quad (\text{A18})$$

where

$$\bar{q}_{lm}(i) = \frac{\sum_{j \in \tilde{N}_b(i)} s(r_{ij}) q_{lm}(j)}{\sum_{j \in \tilde{N}_b(i)} s(r_{ij})}, \quad q_{lm}(i) = \frac{\sum_{j \in N_b(i)} s(r_{ij}) Y_{lm}(\hat{\mathbf{r}}_{ij})}{\sum_{j \in N_b(i)} s(r_{ij})}, \quad (\text{A19})$$

and  $\tilde{N}_b(i) = N_b(i) \cup \{i\}$ . The  $Y_{lm}(\dots)$  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

$$s(r) = \begin{cases} 1, & r < r_{min}; \\ \frac{1}{2} + \frac{1}{2} \cos\left(\pi \frac{r - r_{min}}{r_{max} - r_{min}}\right), & r_{min} \leq r < r_{max}; \\ 0, & r \geq r_{max}. \end{cases} \quad (\text{A20})$$

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

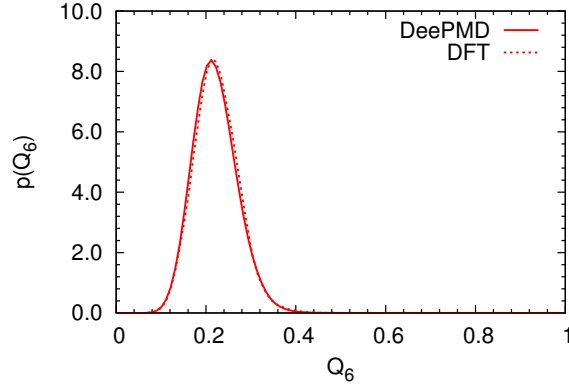


FIG. 9: Probability distribution function of the O-O bond orientation order parameter  $Q_6$

- 
- [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/>