Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics

Article *in* Physical Review Letters · July 2017

DOI: 10.1103/PhysRevLett.120.143001

5 authors, including:

Linfeng Zhang
Princeton University
103 PUBLICATIONS **4,277** CITATIONS
SEE PROFILE

Jiequn Han
Princeton University
72 PUBLICATIONS **4,936** CITATIONS
SEE PROFILE

Han Wang
Institute of Applied Physics and Computational Mathematics
110 PUBLICATIONS **4,409** CITATIONS
SEE PROFILE

Weinan Ee
Princeton University
343 PUBLICATIONS **24,056** CITATIONS
SEE PROFILE

Deep Potential Molecular Dynamics: a scalable model with the accuracy of quantum mechanics

Linfeng Zhang[1], Jiequn Han[1], Han Wang[2,3,*], Roberto Car[1,4], and Weinan E[1,5,6,†]

[1] Program in Applied and Computational Mathematics,
Princeton University, Princeton, NJ 08544, USA
[2] Institute of Applied Physics and Computational Mathematics,
Fenghao East Road 2, Beijing 100094, China
[3] CAEP Software Center for High Performance Numerical Simulation,
Huayuan Road 6, Beijing 100088, China
[4] Department of Chemistry, Department of Physics,
and Princeton Institute for the Science and Technology of Materials,
Princeton University, Princeton, NJ 08544, USA
[5] Department of Mathematics, Princeton University, Princeton, NJ 08544, USA
[6] Center for Data Science and Beijing International Center for Mathematical Research,
Peking University, and Beijing Institute of Big Data Research,Beijing, 100871, China
* Email: wang_han@iapcm.ac.cn
† Email: weinan@math.princeton.edu

ABSTRACT

We introduce a new scheme for molecular simulations, based on a many-body potential and interatomic forces generated by a deep neural network trained with *ab initio* data. We show that the proposed scheme, which we call Deep Potential Molecular Dynamics (DeePMD), 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. Moreover, in a few test cases, DeePMD shows good structural transferability to thermodynamic conditions not included in the original training data.

Although molecular dynamics (MD) has become a popular tool in several disciplines, such as physics, chemistry, biology, and material science, it is still limited by its accuracy and/or cost. *Ab initio* molecular dynamics (AIMD)[3,7,26] has the accuracy of quantum mechanical models based on density functional theory (DFT)[16], but its cost is substantial, limiting applications to systems with hundreds to thousands of atoms. On the other hand, empirical force fields (FF)[13,24,25] make large-scale simulations feasible, but their general transferability is often in doubt, and their construction requires large human effort.

Recent progress has brought in hope that machine learning algorithms [4–6,9,17,20–22] could lead to simulations with DFT accuracy at the cost of empirical potentials. In this context, two of the state-of-the-art methods for MD simulations, the Behler-Parrinello neural network (BPNN)[6] approach and the gradient-domain machine learning (GDML) scheme[9], have shown promising results. However, BPNN requires extensive hand-crafted construction of empirical local symmetry functions, while GDML uses a global input feature, the Coulomb matrix, which is inherently non-scalable.

In this letter, we introduce a molecular dynamics simulation tool, which we call Deep Potential Molecular Dynamics (DeePMD), based on the recently developed Deep Potential method[12]. We demonstrate that DeePMD is a framework capable of modeling the interatomic forces in a variety of systems, ranging from materials in condensed phase, such as liquid water and ice, to isolated organic molecules, such as benzene and aspirin. At the level of model building, DeePMD has no fitting functions other than the network model itself, and requires little human intervention. When DFT data are used to train the model, DeePMD approximates the DFT energy and forces very closely, with errors below chemical accuracy. The structural properties of the AIMD trajectories are reproduced extremely well even at some thermodynamic conditions different from those used for training. The extensive character of the energy is preserved in the DeePMD approach, where energy and forces are sums of local contributions. Thus, the method can be trained on small systems and then applied to large ones. Large-scale applications are further facilitated by the efficiency of the method, which has computational cost that scales linearly with system size.

The construction of the interatomic forces in DeePMD relies on two assumptions: (1) the total energy of the system is the sum of suitably represented "atomic energies", i.e. $E = \sum_i E_i$, i being the index of the atom, and (2) the "atomic energy" E_i is fully determined by the coordinates of the i-th atom and its neighbors within a cut-off radius R_c. These are reasonable assumptions for many condensed matter and molecular systems.

Similar to the procedure adopted in the Deep Potential method[12], the "atomic energy", E_i, is constructed in two steps. First, we set up a local coordinate frame for every atom and its neighbors inside R_c [1]. This allows us to preserve the translational, rotational and permutational symmetries of the environment (Fig. 1). The format adopted for the local coordinate information

[1]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.

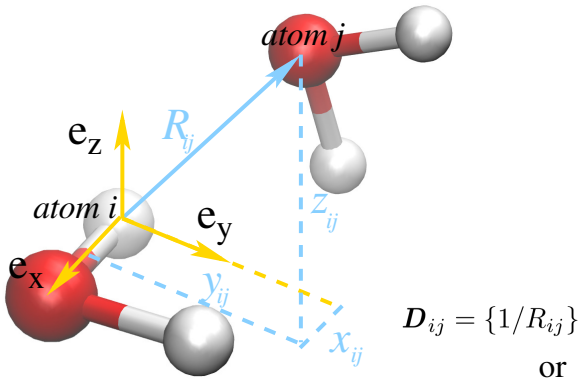

Figure 1: (color online). Schematic plot of the neural network input for the environment of atom i. Atom j is a generic neighbor of atom i. $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$ is the local frame of atom i. $\boldsymbol{e}_x$ is along the O-H bond. $\boldsymbol{e}_z$ is perpendicular to the plane of the water molecule. $\boldsymbol{e}_y$ is the cross product of $\boldsymbol{e}_z$ and $\boldsymbol{e}_x$. (x_{ij}, y_{ij}, z_{ij}) are the Cartesian components of the vector $\boldsymbol{R}_{ij}$ in this local frame. R_{ij} is the length of $\boldsymbol{R}_{ij}$. The neural network input $\boldsymbol{D}_{ij}$ may either contain the full radial and angular information of atom j, i.e. $\boldsymbol{D}_{ij} = \{1/R_{ij}, x_{ij}/R_{ij}^2, y_{ij}/R_{ij}^2, z_{ij}/R_{ij}^2\}$, or only the radial information, i.e. $\boldsymbol{D}_{ij} = \{1/R_{ij}\}$. 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 distance to atom i, i.e. $1/R_{ij}$.

is illustrated in this figure. The information is represented by sets of numbers ($\boldsymbol{D}_{ij}$), which contain either radial data only, or both radial and angular data. The $1/R_{ij}$ factor present in the $\boldsymbol{D}_{ij}$ naturally reduces the weights of the particles that are more distant from atom i. In practice, this representation gives more accurate energies and forces along MD trajectories than representations without the $1/R_{ij}$ factor. Additional details are given in the supplementary materials (SM).

Next, the $\boldsymbol{D}_{ij}$ is standardized to constitute the input of a deep neural network (DNN)[11], which returns E_i as output (Fig. 2). The DNN is a feed forward network, in which data flow from the input layer ($\boldsymbol{D}_{ij}$ of atom i) to the output layer ("atomic energy" E_i), through multiple hidden layers. A hidden layer consists of several nodes that take the input data d_l^{in} from the previous layer and outputs data d_k^{out} to the next layer. We first apply a linear transformation on the input data, i.e., $\tilde{d}_k = \sum_l w_{kl} d_l^{\text{in}} + b_k$. The output data d_k^{out} are then obtained by acting with a non-linear function on the $\tilde{d}_k$, i.e., $d_k^{\text{out}} = \varphi(\tilde{d}_k)$. We use the hyperbolic tangent for the non-linear function φ. This procedure is adopted for all the hidden layers. In the final step, going from the last hidden layer to the final "atomic energies" E_i, only the linear transformation is applied. In all the reported examples, we use 5 hidden layers with decreasing number of nodes

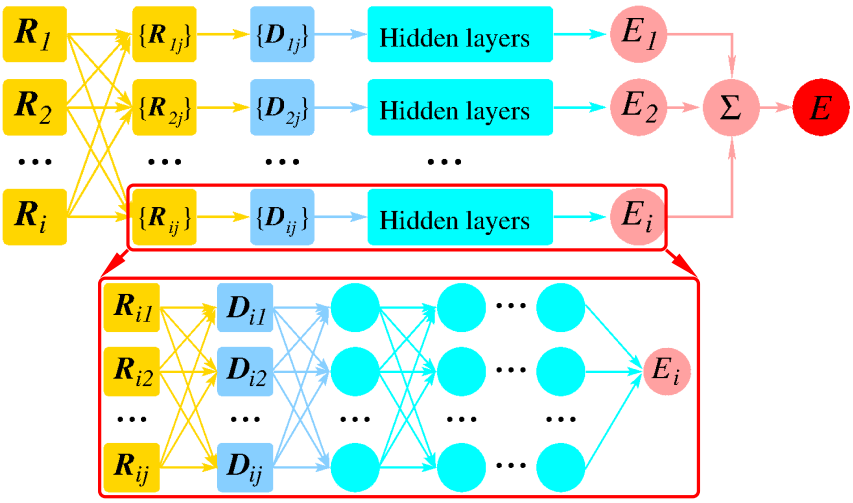

Figure 2: (color online). Schematic plot of the DeePMD model. The frame in the box is the zoom-in of a deep neural network (DNN). The relative positions of all neighbors w.r.t. atom i, i.e. $\{\boldsymbol{R}_{ij}\}$, is first converted to $\{\boldsymbol{D}_{ij}\}$, then passed to the hidden layers to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

per layer, i.e., respectively, 240, 120, 60, 30, and 10 nodes, going from the innermost to the outermost layer.

The parameters w_{kl} and b_k of each layer are weights determined by training, via minimization of the following family of loss functions:

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

$\Delta\epsilon$, $\Delta\boldsymbol{F}_i$ and $\Delta\xi$ are differences between the DeePMD prediction and the training data. ϵ is the total energy divided by the number N of atoms, $\boldsymbol{F}_i$ is the force on atom i, and ξ is the virial tensor Ξ divided by N. 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. In Eq. (9), p_ϵ, p_f and p_ξ are tunable prefactors. In practice, we use the Adam method[15] to update the w_{kl} and b_k of each layer, with a learning rate that exponentially decays with the training step. Based on our experience, the training process is more efficient when the prefactors vary linearly with the learning rate, with p_ϵ and p_ξ increasing and p_f decreasing. In other words, the relative magnitude of the prefactors varies during the training process. When one or two of the three terms, i.e., energy, forces, or virial, are absent from the training data, we set the corresponding prefactor(s) equal to zero. More details in the SM.

In order to test our method, we have applied DeePMD to two classes of systems, i.e., extended

bulk systems and molecules, respectively. As a representative of the first class, we consider water, and use for training liquid configurations of a path-integral AIMD (PI-AIMD) simulation in the NPT emsemble, with pressure $P = 1$ bar and temperature $T = 300$ K. The variable simulation cell contains 64 H_2O molecules with periodic boundary conditions. We adopt $R_c = 6.0$ Å and use the full radial and angular information for the 16 oxygens and 32 hydrogens closest to the atom at the origin, while retaining only radial information for all the other atoms within R_c. We test DeePMD on different liquid and crystalline configurations. In particular, we consider (a) ice Ih at $P = 1$ bar and $T = 273$ K, at the PI-AIMD level, (b) ice Ih at $P = 1$ bar and $T = 330$ K, at the classical AIMD level, and (c) ice Ih at $P = 2.13$ kbar and $T = 238$ K, i.e., the experimental triple point for ice I, II, and III, at the classical AIMD level. All these simulations use a 96-molecule variable cell and include proton disorder. Deuterons replace protons in the simulations (b) and (c). The PBE0+TS[1,23] exchange-correlation functional is adopted in all cases.

As representatives of the second class, we consider the organic molecules benzene, uracil, napthalene, aspirin, salicylic acid, malonaldehyde, ethanol, and toluene, for which classical AIMD trajectories with the PBE+TS functional[18,23] are publicly available [2]. Deep Tensor Neural Network (DTNN)[21] and Gradient Domain Machine Learning (GDML)[9] methods have been tested on these data sets and can serve as benchmarks. In these systems, we set R_c large enough to include all the atoms, and use the full local radial and angular information in input. More details in the SM.

In the following we discuss the performance of DeePMD results according to four criteria: (*i*) generality of the model; (*ii*) accuracy of the energy, of the forces, and of the virial tensor; (*iii*) faithfulness of the trajectories; (*iv*) scalability and computational cost.

Generality. Bulk and molecular systems exhibit very different levels of complexity. Our liquid water sample includes quantum fluctuations. The organic molecules differ in composition and size, and the corresponding data sets include large numbers of conformations. Yet DeePMD produces satisfactory results in all cases, using the same methodology, with the same number of hidden layers and nodes per layer, and the same optimization scheme for the loss function $\mathcal{L}$.

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 ice (Tab. 1), and in terms of the

[2] http://quantum-machine.org/

Table 1: The RMSE of the DeePMD prediction for water and ice 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	1.2	36.8	2.1
ice Ih (a)	1.2	45.4	1.8
ice Ih (b)	1.3	30.9	-
Ice Ih (c)	0.9	29.0	-

Table 2: The MAE of the DeePMD prediction for organic molecules in terms of the energy and the forces. The numbers in the parentheses are the benchmarks provided by Ref. [9].

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)

mean absolute error (MAE) in the case of the organic molecules (Tab. 2). No virial information is used for the molecules. In the water case, the RMSE of the forces is comparable to the accuracy of the minimization procedure used in the original DFT model, in which the allowed error in the forces is less than 10^{-3} *a.u.*. Even though the configurations of the three ices (a, b, and c) are not included in the training, and in spite of the fact that their thermodynamic conditions differ from the liquid, the RMSEs of the forces in the ices are as good as in the liquid. Tab. 1 reports also energy and virial data, the RMSE of which is very good. We should mention, however, that while the average energy of the liquid data is reproduced extremely well by DeePMD, the average energies of the ice structures are shifted by varying amounts depending on the structure. The largest shift ($\sim$10 meV/H_2O) is observed for ice Ih at the triple point, which is the thermodynamic condition that differs mostly from the training data. In the case of the molecules, the results for energy and forces are generally better than the GDML benchmark.

MD trajectories. In the case of water and ice, we perform path-integral/classical DeePMD simulations at the thermodynamic conditions of the original models, using the i-PI software [8]. We obtain almost the same density $\bar{\rho}$ and radial distribution functions (RDFs) of the original model. In the liquid the average energy $\bar{E}$ is in excellent agreement with the original data. The results are summarized in Tab. 3. The RDFs of the quantum trajectories for water and ice are

Table 3: The equilibrium energy $\bar{E}$ and the equilibrium density $\bar{\rho}$ of water and three ices (a, b, c), with DeePMD and AIMD. The numbers in parentheses are the AIMD results.

System	$\bar{E}$[eV/H_2O]		$\bar{\rho}$[g/m^3]	
liquid water	-467.682	(-467.681)	1.015	(1.012)
ice Ih (a)	-467.749	(-467.747)	0.966	(0.966)
ice Ih (b)	-468.048	(-468.055)	0.950	(0.950)
ice Ih (c)	-468.094	(-468.102)	0.985	(0.986)

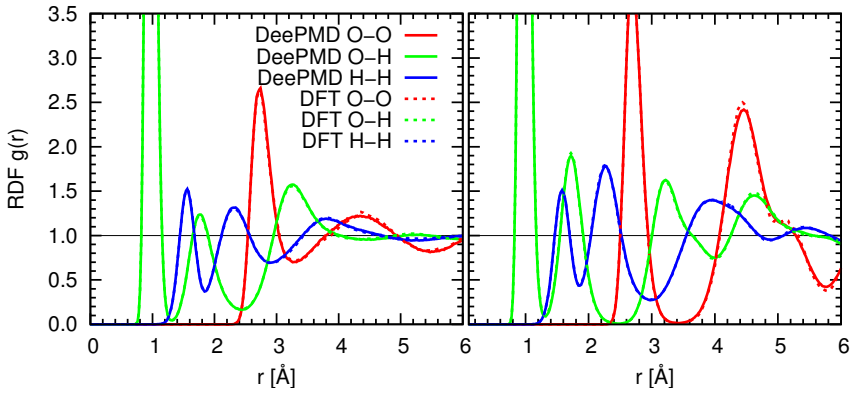

Figure 3: (color online) Comparison between the DeePMD and the PI-AIMD RDFs of liquid water (left) and ice Ih (a) (right).

shown in Fig. 3. Additional RDFs (for ice (b) and (c)) are reported in the SM. In the case of the molecules, we perform DeePMD at the same temperature of the original data, with a Langevin thermostat with a damping time $\tau = 0.1$ ps. The corresponding distributions of interatomic distances (Fig. 4) are very close to the original data.

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 directly applied to much larger systems. The total computational cost of DeePMD scales linearly with the number of atoms. Moreover, DeePMD can be easily parellelized 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 AIMD and empirical force field TIP3P[14] simulations in units of CPU core seconds/step/molecule.

It should also be stressed that in our approach the training cost is quite small, typically ∼3 hours for each one of the molecules, and ∼18 hours for bulk liquid water, on a Thinkpad P50 laptop computer with an Intel Core i7-6700HQ CPU.

A few issues require further investigation. First, in the current implementation there are small discontinuities in the potential energy surface, due to the sharp cutoff, the fixed number

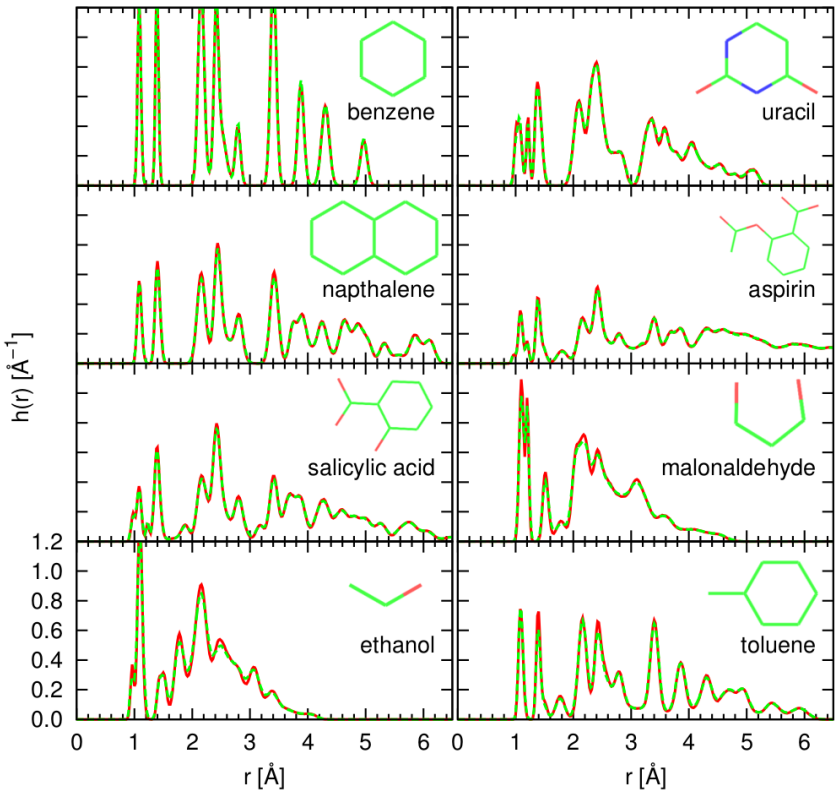

Figure 4: (color online) Interatomic distance distributions of the organic molecules. The solid lines denote the DeePMD results. The dashed lines denote the AIMD results. The interatomic distance distribution is defined by $h(r) = \langle 2/\big[N(N-1)\big] \sum_{i<j}^{N} \delta(r - |\boldsymbol{R}_i - \boldsymbol{R}_j|) \rangle_{P,T}$.

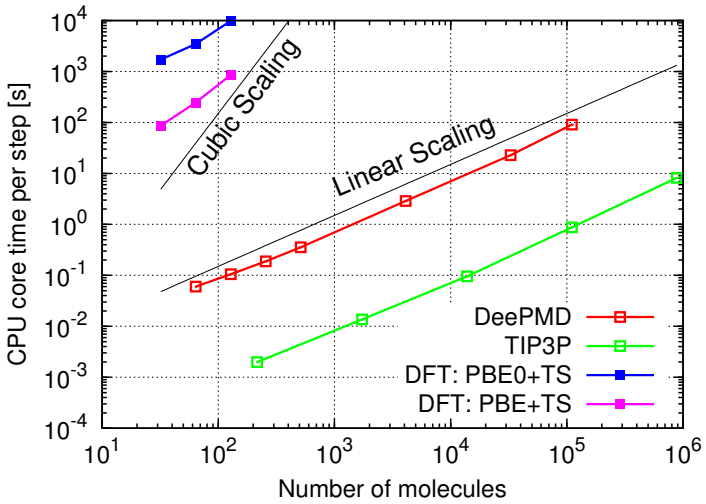

Figure 5: Computational cost of MD step *vs.* system size, with DeePMD, TIP3P, PBE+TS and, PBE0+TS. All simulations are performed on a Nersc Cori supercomputer with the Intel Xeon CPU E5-2698 v3. The TIP3P simulations use the Gromacs codes (version 4.6.7)[19]. The PBE+TS and PBE0+TS simulations use the Quantum Espresso codes[10].

of atoms whose radial and angular information is used, the discontinuous changes of the local frames and of the local atomic lists, associated to the sorting procedure [3]. These discontinuities have negligible effects in canonical simulations, but cause a small energy drift in microcanonical runs. Another issue is the shift in the average energy observed in DeePMD simulations of structures not included in the training set. While irrelevant for structural properties such as the RDFs, these shifts are important for the relative stability of different phases (like, e.g., liquid and solid water). Finally, long-range Coulomb effects are not treated explicitly in the current implementation, although implicitly they are present in the training data. Explicit treatment of long-range Coulomb effects may be necessary in some applications and deserves further study. A possible way of including explicitly these effects in neural network models was discussed in Ref. [2].

In summary, DeePMD is a new molecular simulation tool that can successfully address the dilemma of accuracy *vs.* cost that has confronted the molecular simulation community for a long time. It is general, scalable, and has accuracy comparable to that of the training data. With this tool, one can use highly accurate *ab initio* data on a relatively small system for training, and then apply the resulting DeePMD model to much larger systems. In the present implementation, the computational cost of DeePMD is somewhat larger than that of MD with empirical potentials but is several orders of magnitude smaller than that of *ab initio* simulations.

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. 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.

References

[1] Adamo, C. and Barone, V., The Journal of Chemical Physics **110**, 6158 (1999).

[2] Artrith, N., Morawietz, T., and Behler, J., Physical Review B **83**, 153101 (2011).

[3] some of the discontinuities could be trivially eliminated by adopting smooth cutoffs, etc., but those related to the discontinuous changes in the ordered lists are more severe and deserve further study.

[3] Barnett, R., Landman, U., Nitzan, A., and Rajagopal, G., The Journal of Chemical Physics **94**, 608 (1991).

[4] Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G., Physical Review Letters **104**, 136403 (2010).

[5] Behler, J., The Journal of Chemical Physics **145**, 170901 (2016).

[6] Behler, J. and Parrinello, M., Physical Review Letters **98**, 146401 (2007).

[7] Car, R. and Parrinello, M., Physical Review Letters **55**, 2471 (1985).

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

[9] Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., and Müller, K.-R., Science Advances **3**, e1603015 (2017).

[10] Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G. L., Cococcioni, M., Dabo, I., Dal Corso, A., de Gironcoli, S., Fabris, S., Fratesi, G., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., Martin-Samos, L., Marzari, N., Mauri, F., Mazzarello, R., Paolini, S., Pasquarello, A., Paulatto, L., Sbraccia, C., Scandolo, S., Sclauzero, G., Seitsonen, A. P., Smogunov, A., Umari, P., and Wentzcovitch, R. M., Journal of Physics: Condensed Matter **21**, 395502 (19pp) (2009).

[11] Goodfellow, I., Bengio, Y., and Courville, A., *Deep learning* (MIT Press, 2016).

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

[13] Jorgensen, W., Maxwell, D., and Tirado-Rives, J., Journal of the American Chemical Society **118**, 11225 (1996).

[14] Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., and Klein, M. L., The Journal of Chemical Physics **79**, 926 (1983).

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

[16] Kohn, W. and Sham, L. J., Physical Review **140**, A1133 (1965).

[17] Morawietz, T., Singraber, A., Dellago, C., and Behler, J., Proceedings of the National Academy of Sciences , 201602375 (2016).

[18] Perdew, J. P., Burke, K., and Ernzerhof, M., Physical Review Letters **77**, 3865 (1996).

[19] Pronk, S., Páll, S., Schulz, R., Larsson, P., Bjelkmar, P., Apostolov, R., Shirts, M., Smith, J., Kasson, P., van der Spoel, D., Hess, B., and Lindahl, E., Bioinformatics , btt055 (2013).

[20] Rupp, M., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A., Physical Review Letters **108**, 058301 (2012).

[21] Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A., Nature Communications **8**, 13890 (2017).

[22] Smith, J. S., Isayev, O., and Roitberg, A. E., Chemical Science **8**, 3192 (2017).

[23] Tkatchenko, A. and Scheffler, M., Physical Review Letters **102**, 073005 (2009).

[24] Vanommeslaeghe, K., Hatcher, E., Acharya, C., Kundu, S.and Zhong, S., Shim, J., Darian, E., Guvench, O., Lopes, P., Vorobyov, I., and Mackerell Jr., A., Journal of Computational Chemistry **31**, 671 (2010).

[25] Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A., and Case, D. A., Journal of Computational Chemistry **25**, 1157 (2004).

[26] Wentzcovitch, R. M. and Martins, J., Solid State Communications **78**, 831 (1991).

A Supplementary Materials

A.1 Data Description

A.1.1 water and ice

The data used for training and/or testing are extracted from the AIMD simulations summarized in Tab. 4. All the simulations adopt a time step of 0.48 fs. The PI-AIMD simulations use the CPMD codes of Quantum Espresso [4] for the DFT part, 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[6] for thermalization. The Parrinello-Rahman technique[7] for variable cell dynamics is adopted in all cases. 95000

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

Table 4: 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 (a)	path integral	96	1.0	273	1.5
ice (b)	classical	96	1.0	330	7.0
ice (c)	classical	96	2.13k	238	7.0

snapshots (from 105000 total snapshots), randomly selected along the liquid water trajectory, are used to train the DeePMD model.

A.1.2 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 molecules, 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.

A.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.2.1 network input data

We consider a system with 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. We consider the neighbor list $\mathcal{N}(i)$ to be sorted according to the scheme illustrated in Fig. 1 of this paper. 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 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 use triplets of closest atoms, independently of their species, to define the local frame. Thus, in all the water cases, we use the three atoms belonging to a single

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,

$$\mathcal{R}(\boldsymbol{R}_{ia(i)}, \boldsymbol{R}_{ib(i)}) = \begin{pmatrix} \boldsymbol{e}[\boldsymbol{R}_{ia(i)}] \\ \boldsymbol{e}[\boldsymbol{R}_{ib(i)} - (\boldsymbol{R}_{ia(i)} \cdot \boldsymbol{R}_{ib(i)} \boldsymbol{R}_{ia(i)}] \\ \boldsymbol{e}[\boldsymbol{R}_{ia(i)} \times \boldsymbol{R}_{ib(i)}] \end{pmatrix}^T , \tag{2}$$

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

$$\boldsymbol{R}'_{ij} = \{x'_{ij}, y'_{ij}, z'_{ij}\} = \{x_{ij}, y_{ij}, z_{ij}\}\, \mathcal{R}(\boldsymbol{R}_{ia(i)}, \boldsymbol{R}_{ib(i)}), \tag{3}$$

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

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

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 labelled 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_{ij}^{α} 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 shifting operations, we divide by 0.01 Å^{-1} the D_{ij}^{α} with standard deviation smaller than 0.01 Å^{-1}. To simplify the notation, we still use the same notation for the standardized D_{ij}^{α}.

A.2.2 the energy, the forces, and the virial tensor

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

$$E_i = N_{\boldsymbol{w}(i)}(\{D_{ij}^{\alpha}\}_{j\in\mathcal{N}(i),\alpha}) \tag{4}$$

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. Atoms of the same species have the same $\boldsymbol{w}$.

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

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

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

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.,

$$\Xi_{\alpha\beta} = -\frac{1}{2}\sum_{\gamma}\frac{\partial E}{\partial h_{\alpha\gamma}}h_{\gamma\beta}, \tag{5}$$

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:

$$\Xi_{\alpha\beta} = -\frac{1}{2}\sum_{i,j} x_{\alpha}^{(i,j)} f_{\beta}^{(i,j)}. \tag{6}$$

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

$$x_{\alpha}^{(i,j)} = x_{\alpha}^{(i)} - x_{\alpha}^{(j)}. \tag{7}$$

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

$$f_{\beta}^{(i,j)} = -\frac{\partial E_i}{\partial_{x_j^{\beta}}}. \tag{8}$$

This completes the definition of all the quantities needed for training and MD simulations.

A.2.3 Training Details

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

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

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=1.0×10^{-8}, which are the default settings in TensorFlow. The learning rate r_l decays exponentially with the global step:

$$r_l = r_{l0} d_r^{-c_s/d_s}, \tag{10}$$

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:

$$p = p_{limit}(1 - \frac{r_l}{r_{l0}}) + p_{start}(\frac{r_l}{r_{l0}}), \tag{11}$$

in which p_{start} is the prefactor at the beginning, 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:

$$\begin{cases} p_{estart} = 1, & p_{elimit} = 400; \\ p_{fstart} = 1000, & p_{flimit} = 1, \end{cases} \tag{12}$$

for both water and molecules, and

$$\begin{cases} p_{vstart} = 1, p_{vlimit} = 400, & \text{for liquid water;} \\ p_{vstart} = 0, p_{vlimit} = 0, & \text{for molecules.} \end{cases} \tag{13}$$

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. Since adding the forces and/or the virial tensor provides a strong regularization of the network and makes training more efficient, Batch Normalization techniques are

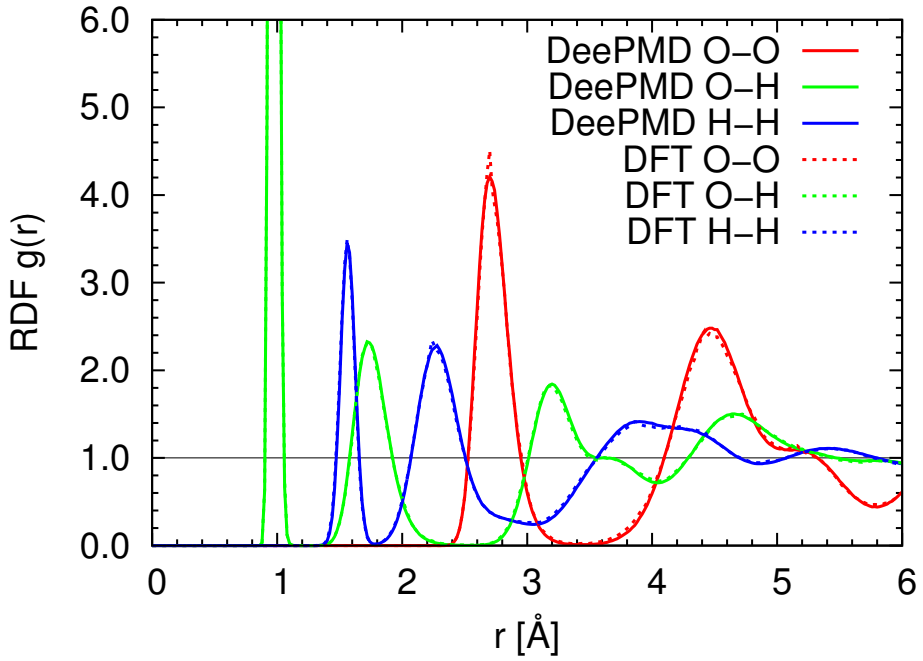

Figure 6: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (b).

not necessary within the DeePMD framework.

A.2.4 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 100 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.

A.3 Additional Results

The RDFs for ice Ih (b) and (c) are reported in Figs. 6 and 7, respectively.

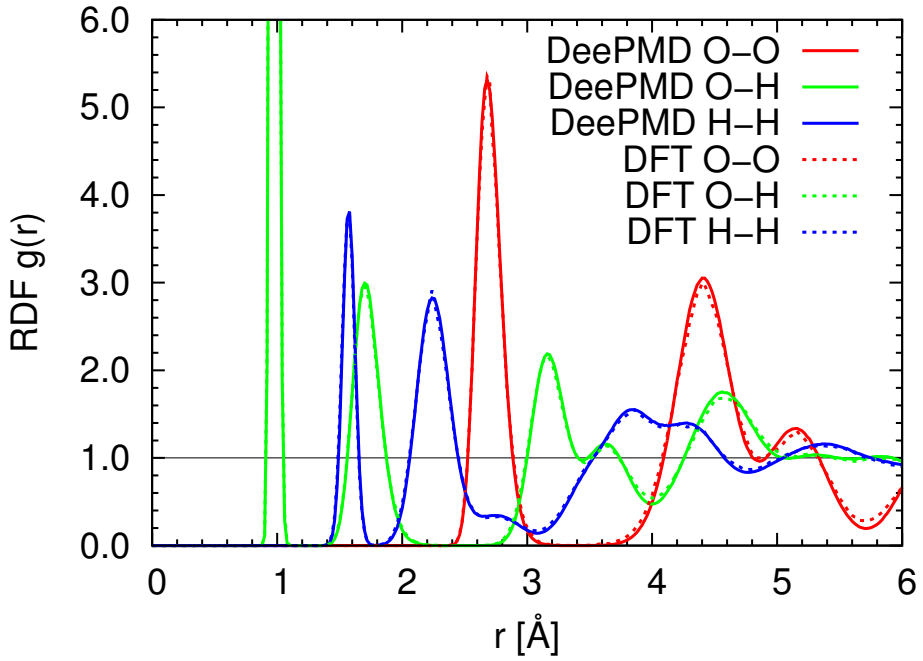

Figure 7: The comparison between the DeePMD RDFs and the AIMD RDFs of ice Ih (c).

Supplementary References

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

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