The Journal of Chemical Physics

RESEARCH ARTICLE | JANUARY 25 2023

Learning pair potentials using differentiable simulations

Wujie Wang ; Zhenghao Wu ; Johannes C. B. Dietschreit ; Rafael Gómez-Bombarelli

J. Chem. Phys. 158, 044113 (2023)
https://doi.org/10.1063/5.0126475

AIP Publishing

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

Learning pair potentials using differentiable simulations

Cite as: J. Chem. Phys. 158, 044113 (2023); doi: 10.1063/5.0126475
Submitted: 15 September 2022 • Accepted: 15 December 2022 •
Published Online: 25 January 2023

Wujie Wang,[1] Zhenghao Wu,[2] Johannes C. B. Dietschreit,[1] and Rafael Gómez-Bombarelli[1,a)]

AFFILIATIONS

[1] Department of Materials Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, Massachusetts 02139, USA

[2] Eduard-Zintl-Institut für Anorganische und Physikalische Chemie, Technische Universität Darmstadt, Alarich-Weiss-Str. 8, 64287 Darmstadt, Germany

[a)] **Author to whom correspondence should be addressed:** rafagb@mit.edu

ABSTRACT

Learning pair interactions from experimental or simulation data is of great interest for molecular simulations. We propose a general stochastic method for learning pair interactions from data using differentiable simulations (DiffSim). DiffSim defines a loss function based on structural observables, such as the radial distribution function, through molecular dynamics (MD) simulations. The interaction potentials are then learned directly by stochastic gradient descent, using backpropagation to calculate the gradient of the structural loss metric with respect to the interaction potential through the MD simulation. This gradient-based method is flexible and can be configured to simulate and optimize multiple systems simultaneously. For example, it is possible to simultaneously learn potentials for different temperatures or for different compositions. We demonstrate the approach by recovering simple pair potentials, such as Lennard-Jones systems, from radial distribution functions. We find that DiffSim can be used to probe a wider functional space of pair potentials compared with traditional methods like iterative Boltzmann inversion. We show that our methods can be used to simultaneously fit potentials for simulations at different compositions and temperatures to improve the transferability of the learned potentials.

I. INTRODUCTION

Molecular simulation has been an invaluable technique in various fields of modern science, including chemistry, physics, and materials science.[1–4] Ideally, an atomistic computer simulation would combine the accurate description of the Born–Oppenheimer potential energy surface (PES) with the treatment of nuclear motion, both at the quantum mechanics level. However, this is computationally infeasible for most of the relevant molecular sizes and time scales. It is typically necessary to approximate the PES with simpler and faster surrogate functions, such as force fields or interatomic potentials. Therefore, the development of accurate and computationally efficient force fields has become one of the most important topics in molecular simulations.[5–7]

Covalent bonds in organic molecules are typically described in force fields as harmonic or polynomial expressions in short-range many-body terms (bonds, angles, and torsions), while numerous types of descriptions have been proposed for noncovalent interactions.[8,9] Of all these variants, the pairwise potential is most commonly used in molecular simulations, as it has been shown to be sufficient to capture significant molecular mechanics, and is also the most computationally cost-effective way to describe noncovalent interactions between particles.[9] For example, rigid water models with pairwise additive PESs such as extended simple point charge (SPC/E)[10] and TIP4P[11] recover many thermodynamic properties, e.g., densities and heat capacities, which are consistent with experimental measurements across a wide range of temperatures. These models with pairwise additive PESs are efficient in terms of computational cost and scalability.[12] It has been noted that the many-body expansion of PESs is appreciated for its role in accurately predicting behaviors or properties, such as structures or phase transitions of water, which the pairwise additive PES cannot predict.[9,13] Furthermore, many-body terms seem to play an important role when coarse graining proteins to single beads per residue.[14] However, these potential formulations that carry the many-body effect are usually not as efficient; depending on the size of the system, they are usually

10–1000 times slower to simulate as their pairwise counterparts.[14,15] In this work, we focus on pairwise interactions due to their simplicity and visualize ability.

In addition to all-atom potentials, the derivation of effective pair potentials also plays an important role in the efficient simulation of coarse-grained systems.[16–18] Various variational frameworks have been developed to derive coarse-grained potentials. Based on the idea of force matching,[19] Izvekov and Voth developed a force-based coarse-graining method, i.e., multiscale coarse-grained method (MS-CG).[20] The MS-CG method aims to reproduce the many-body potential of the mean force (PMF) of the fine-grained system (FG) in the mapped CG representation by matching the force acting on the CG bead.[21] This framework can also be extended to different architectures of force fields, e.g., neural force fields, optimized with stochastic gradient descent.[22,23] However, it is a known issue that the potentials learned from force matching do not necessarily recover target structural distributions due to missing cross-correlations between degrees of freedom.[24] The use of neural force fields naturally incorporates many-body correlation terms, but they often suffer from instabilities in dynamics and often fail to generate robust simulation trajectories.[23,25] Beyond force matching, Chaimovich and Shell developed a series of coarse-graining approaches using relative entropy to measure coarse-graining information loss.[26] The standard relative entropy framework uses the Newton–Raphson algorithm as the solver, which, however, is computationally expensive for optimizing models such as deep neural networks that have a large number of parameters.[27] There is another class of structure-based coarse-graining method, which does not require atomistic position or force data and directly aims at reproducing structural distributions, of which iterative Boltzmann inversion (IBI) is the most notable.[28] IBI constructs potentials for a CG model by iteratively correcting errors in simulated radial distribution function until convergence.[28] It has been widely employed for coarse-graining various soft-matter systems such as organic liquids,[29] ionic liquids,[30] and polymers.[29] IBI does not necessarily require reference all-atom simulations. However, its application is limited to tabulated pair potentials and does not generalize to other forms of potential.

Most coarse-graining approaches only aim at reproducing statistics in a single thermodynamic state, so the developed potentials are rarely transferable. Several approaches have been proposed to improve the transferability of the coarse-graining frameworks mentioned in previous sections. For example, Mullinax and Noid proposed an extended ensemble method in the MS-CG framework,[31] in which datasets from multiple equilibrium ensembles are collected to develop CG potentials that reproduce the potential of mean force of FG systems in these ensembles. This approach requires a considerable amount of atomistic force data, which hinders its application.

CG models created by the IBI method can reproduce the structural correlations of the FG model at a single thermodynamic state, but they generally do not produce potentials that are transferable to a wide range of thermodynamic states with different temperatures and compositions. It is possible to derive more transferable CG potentials using multi-state data based on IBI, which can through selecting the right weights tune other system properties.[32] However, actively optimizing targets beyond the RDF is not possible by construction with IBI.

In recent years, machine learning (ML) has emerged as an important tool for molecular modeling. Much of the success has been based on the application of differentiable programming, which enables the exact generation of gradient programs. The process of generating the gradient program is termed Automatic Differentiation (AD). Although emerging as a relatively new tool, AD has shown great progress in many domains of computational science, such as quantum computing[33] and fluid dynamics.[34] For the modeling of PES, AD has transformed the fitting of force fields from first-principle calculations: once a forward program that transforms nuclear coordinates to a scalar (energy) is constructed, the gradient (force) program is automatically generated with AD. The generated force field is also trainable with respect to the data, enabling parameterization of ML potentials with near *ab initio* accuracy.[35]

Differentiable programming is applicable to many controlled iterative mathematical procedures in the physical sciences, ranging from learning exchange correlation functionals for Density Functional Theory (DFT) calculations from data[36,37] or fine-tuning molecular basis sets.[38] This requires differentiating through mathematical procedures such as fixed-point iteration[39] and eigendecomposition.[40] Similarly, differentiation operations of molecular simulations, as solutions to some ordinary differential equation (ODE), can also be defined.[41] Recently, many applications have been proposed using differentiable molecular dynamics (DiffSim).[42–46] Specifically, recent work on DiffSim suggests a differentiable top-down approach to learning interatomic potentials directly based on macroscopic observations from experiments or simulations. Compared with previously proposed machine learning potentials that are trained from the bottom-up, DiffSim enables experimentally informed parameterization of force fields. However, to improve fitting flexibility, these methods utilize neural network (NN) potentials based on graph neural networks, which have limited interpretability and are less scalable for large-scale simulations. Another method of machine learning-assisted top-down parameterization involves directly predicting force field parameters from macroscopic structure correlation using a machine-learned function to bypass the inverse problem.[47] This approach requires abundant labeled simulation data and is limited to a simple functional form of Lennard-Jones type, which has limited capacity to encode complex structural correlations.

In this work, we propose simulation-informed parameterization of pair potentials using DiffSim. Compared with structure-based methods like IBI, our approach can be used to optimize flexible functional forms and can be easily configured to fit multiple systems simultaneously; compared with force-matching-based methods, our method does not require atomistic position or force data and aims at directly reproducing the target distributions. First, we introduce our method and training protocols. For numerical examples, we demonstrate our method on simple liquid systems where ground truth potentials are known. We show that our method is good at obtaining a diverse set of pair potentials that reproduce the target liquid structure. To demonstrate the flexibility of the method, we show that the model can learn pair potentials for binary mixture systems involving multiple interactions and compositions. Our results indicate that fitting multiple states simultaneously helps to improve the transferability of learned potentials. Furthermore, our method can be used to learn coarse-grained potentials where there

is no ground truth for pairwise potentials. We show that the learning of interactions can be combined with flexible functional forms that involve temperature dependence for improved transferability (Fig. 1).

II. METHOD

A. Observable-based differentiable simulations

Differentiable programming is a programming paradigm in which the gradient of a mathematical procedure can be automatically synthesized. In deep learning, automatically computed gradients allow gradient-based optimization using "backpropagation" to train deep neural networks. As universal approximators, neural networks of various architectures have been used as a drop-in module for optimization and learning. However, without the appropriate incorporation of inductive biases, many NN models tend to overfit the data and often fail to generalize.[48] With the development of differentiable algorithms, many sampling and optimization procedures can now be directly incorporated into the machine learning framework. These structures serve as inductive biases to regularize learning, improve data efficiency, and improve model interpretability.[49] Machine learning of dynamical systems and differentiable equations is a domain where differentiable sampling operations are used as strong inductive biases.[50–52]

In molecular dynamics, the simulation of systems of many particles requires integrating per-particle state variables in time following the governing equations of motion. Although continuous, the computational solution of state variable dynamics is solved in discrete time steps, involving the composition of differentiable mathematical procedures. With AD, the computation of observables from molecular simulations is, therefore, end-to-end differentiable. Specifically, given a defined forward computational program, the AD framework traces the computation backward and recursively applies chain rules as vector Jacobian products. This requires the implementation of the accompanying gradient function for every primitive operation used for the forward computation. AD can be applied to arbitrarily complex numerical programs, as long as the gradient of each elementary step can be computed. Auto-differentiable code has been implemented in many modern numerical computing packages.[53–55]

The evolution of atomic coordinates in MD is typically obtained by integrating the equations of motion through a discretized update rule of the following form,

$$x_t = F_\theta(x_{t-1}, v_{t-1}, \eta_{t-1})dt + x_{t-1}, \tag{1}$$

where x is the coordinate, v is the velocity, η is the bath variable, θ represents the set of learning parameters, and F is the discretized update procedure. The simulation observables O are usually extracted from the trajectories. Examples of such observables include radial distribution functions (RDF), velocity autocorrelation functions, etc. To learn or control the simulated observables, a scalar loss function L must be defined to quantify the discrepancy between the simulated and target observables. The loss function L needs to be minimized using gradient descent with gradient on θ. We first consider the computation of the gradient on x, or the adjoint state, at each time step,

$$\begin{aligned}\frac{dL}{dx_t} &= \frac{\partial L}{\partial x_t} + \frac{dL}{dx_{t+1}}\frac{\partial x_{t+1}}{\partial x_t} \\ &= \frac{\partial L}{\partial x_t} + \frac{dL}{dx_{t+1}} + \frac{dL}{dx_{t+1}}\frac{\partial F_\theta}{\partial x_t}dt.\end{aligned} \tag{2}$$

With $\frac{dL}{dx_t}$ at each step t, the gradient of θ can be accumulated using the chain rule,

$$\frac{dL}{d\theta} = \sum_t^\intercal \frac{dL}{dx_t}\frac{dx_t}{d\theta} = \sum_t^\intercal \frac{dL}{dx_t}\frac{dF_\theta(x_{t-1}, \ldots)}{d\theta}dt. \tag{3}$$

Alternatively, one can treat Eq. (3) as a dynamical system solved backward in time.[41] With the gradient accumulated from MD simulations for $\intercal$ steps, gradient-based optimization can be applied to minimize the observable optimization objective L until convergence.

1. Differentiable computation of radial distributions functions

The goal of our method is to learn pair interactions from RDFs, which are treated as our observable O. RDF is defined as

$$g(r) = \frac{V}{N^2 4\pi r^2}p(r), \tag{4}$$

where V is the volume, N is the number of particles, and r is the radial distance from a target particle; $p(r)$ is the histogram of pair distances. To propagate the gradient from O to the model parameters, $p(r)$ must be differentiable, while the operation to

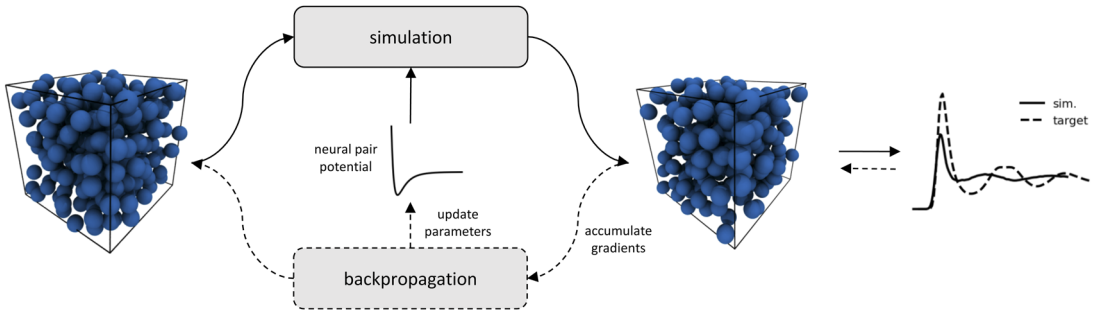

FIG. 1. A schematic diagram of our learning framework.

obtain a histogram is not differentiable. To make the computation of $g(r)$ differentiable, we apply the kernel density trick with Gaussian kernels to make the procedure for computing histograms differentiable.[51] By definition,

$$p(r) = \sum_{i \neq j} \frac{\delta(r_{ij} - r)}{\sum_{i \neq j} 1}, \tag{5}$$

where r_{ij} is the distance between the particle i and j, and $\delta(\cdot)$ is the Dirac Delta function. To estimate $\delta(\cdot)$ in a differentiable way, we approximate the probability density of two particles that are μ_k apart in distance as

$$p(\mu_k) \approx \sum_{i \neq j} p_k(r_{ij}) = \sum_{i \neq j} \frac{e^{-(\mu_k - r_{ij})^2/d}}{\sum_{k'=0}^{k=K-1} e^{-(\mu_{k'} - r_{ij})^2/d}}, \tag{6}$$

where $d = \mu_{k+1} - \mu_k$ is the size of the histogram bin and K is the total number of evenly spaced Gaussian kernels. The summation in the denominator ensures that the histogram is normalized over $[\mu_0, \mu_{K-1}]$. This approximation provides a density estimate over binned domains. Similar methods with triangular kernels have also been proposed before.[56] As $d \to 0$, the Gaussian kernel function approximates $\delta(\cdot)$.

2. Design for neural pair potentials

We propose a design for the neural network $NN : \mathbb{R} \to \mathbb{R}$ with a parameter set θ as an approximation function for pair potentials. We choose neural networks (NN) to parameterize pair potentials because NNs are universal approximators[57] and can be easily stacked in a model for end-to-end training. The approach is also applicable to parameterizing fixed functional forms such as Lennard-Jones or other choices. Given a distance r_{ij} within a cutoff distance r_{cut}, we expand the distance with a Gaussian basis to obtain a distance feature,

$$e_k(r_{ij}) = e^{-\frac{(\mu_k - r_{ij})^2}{d_k}}, \tag{7}$$

where the initial d_k is learnable with the initial value set as r_{cut}/K and K is the number of Gaussian bases used to parameterize r_{ij}. The K dimensional distance feature is then fed into a standard multilayer perceptron (MLP) with n_{layer} hidden layers and W hidden nodes per layer to parameterize a scalar energy output. We term our neural pair potential u_θ with θ indicating the set of learnable parameters that include d_k and weights and biases in MLP. AD can be applied to generate the force on an individual particle i to be used for simulation. The force program is also differentiable to receive gradient signals to adjust the parameter set θ. In addition to the MLP-based pair potential, we also include an unlearnable prior pair potential U_{prior} to guide the initial simulations to explore reasonable regions of the configuration space. For u_{prior}, we simply use a repulsive potential of the form $\sum_{i,j} (\frac{\sigma}{r_{ij}})^c$ where c is the repulsive exponent, which is also treated as a hyperparameter. The total pair potentials used for DiffSim take the form,

$$U_\theta(x) = \sum_m \sum_{i,j \in \mathscr{P}_m} u_\theta^m(r_{ij}) + u_{prior}(r_{ij}), \tag{8}$$

where m indicates the type of pair interactions and $\mathscr{P}_m$ indicates the set of atom pairs that are governed by the pair potential of the same type. In case faster convergence is needed, we also perform an extra pre-training step to set the pair potentials with the direct Boltzmann inversion[58] so that $u_\theta^m(r_{ij}) + u_{prior}(r_{ij}) \approx -k_B T \ln g^m(r)$ with $g^m(r)$ being the target RDF where k_B and T are the Boltzmann constant and the absolute temperature, respectively.

3. Learning protocols

In this section, we introduce our protocols for learning neural pair potentials. The learning protocol follows the concept of learning-to-simulate,[59] with the simulator performing the coupled task of sampling data points and learning. We first initialize the MD systems as crystal structures (FCCs or BCCs) at a target density. We then apply DiffSim to evolve the system for T steps and compute the observable of interest. The examples used in this work are simulated in the canonical ensemble (NVT) using the Nosé–Hoover chain integrator.[60] To learn pair potentials, we are interested in learning a U_θ that produces a target RDF $g_{\text{ref.}}(r)$. We term the RDF obtained from DiffSim as $g_{\text{sim.}}(r)$. We then want to minimize the L2 loss between targets and simulated RDFs,

$$L = \sum_m \int_r (g^m_{\text{sim.}}(r) - g^m_{\text{ref.}}(r))^2 dr, \tag{9}$$

where m again indicates the type of pair potential. With DiffSim, we can then obtain gradients on the parameters according to Eq. (3). The simulation explores the configuration space with an initial potential and accumulates gradients based on the simulated trajectories.

Gradients can be fed into a gradient-based optimizer, such as Adam,[61] to update U_θ. We simulate and update the pair potentials for N_{epoch} epochs until the learning converges. Because differentiable simulations can be susceptible to exploding or vanishing gradients,[62,63] we tend not to evolve the system too long before each gradient updates. In practice, we observe poor learning outcomes with simulation steps that are more than 300 steps. We applied small learning rates to perturb the simulation slightly between forward and backward passes. The learning rate is also scheduled to decrease if it reaches a plateau during learning. The decreasing learning rate facilitates fine-tuning of the learned potentials when the sampled configurations produce an observation close to the target. To report the equilibrium RDF, we simulate our example systems sufficiently long to reflect the true RDF governed by the learned pair interactions. In our experiments, we choose T between 100 and 300 to ensure a stable gradient calculation and to provide sufficient time for the system to evolve based on the updated force field.

B. Iterative Boltzmann inversion

We compared our proposed learning protocol with iterative Boltzmann inversion (IBI), a popular structure-based method to determine effective pair potentials from RDF.[28] IBI is the iterative version of the Boltzmann inversion (BI).[58] BI is a simple method based on the fact that the distribution of an independent degree of freedom x corresponds to a Boltzmann distribution in a canonical ensemble,

$$p(x) \propto \exp\left(-\frac{u(x)}{k_B T}\right), \tag{10}$$

where $p(x)$ is the normalized distribution. The potential $u(x)$ can be obtained by inverting the above equation,

$$u(x) = -k_B T \log g_{\text{ref.}}(x). \tag{11}$$

The potential obtained is in the form of tabulated pair potentials. Such a direct inversion ignores the indirect force contribution to the pair statistics, and therefore, introduces errors in reproducing the target RDF.[28,58] As a natural extension of BI, IBI iteratively corrects the potential using the difference between the target distribution and the distribution obtained from the trial potential energy function,

$$u_{t+1}(x) = u_t(x) + \alpha \Delta u_t, \tag{12}$$

and

$$\Delta u_t = k_B T \ln\left(\frac{g_t(x)}{g_{\text{ref.}}(x)}\right), \tag{13}$$

where u_t is the effective potential at iteration t and α is the size of the update step. Here, $g_t(x)$ is the RDF simulated with u_t, and $g_{\text{ref.}}(x)$ represents the RDF calculated from a reference system. Since IBI is only compatible with tabulated pair potentials, it cannot be used to optimize a broader class of potentials such as neural networks and pair potentials with explicit thermodynamic dependence, which are used in this work. Practically, IBI also suffers from convergence issues, as in some cases hundreds of iterations are required to converge.[64,65] In our work, we use IBI as a baseline comparison, and all CG potentials are obtained using the Versatile Object-Oriented Toolkit for Coarse-Graining Applications (VOTCA).[66]

III. RESULTS

A. Learning simple pair potentials

As a first example, we apply our method to simple Lennard-Jones (LJ) systems at different densities. We obtain target RDFs for LJ particles simulated at three densities $\rho = (0.3, 0.5, 0.7)$ with the temperature set at 1.0, and we use the obtained RDFs as the optimization target for our DiffSim protocol. We obtain the target RDFs by simulating the systems for 500 000 steps after the initial equilibration of 100 000 steps. We use a time step of 0.005. To test the transferability of learned potentials at different system densities, we first learn pair potentials at a single density and use the learned potential to simulate the system with the other two densities that are not used for learning. The learning results are presented in Fig. 2, which also includes a comparison with the potentials obtained using IBI. Our method shows an accurate fitting of the RDFs for the dedicated densities, but the level of transferability varies. The potential learned at the low density of $\rho = 0.3$ demonstrates excellent transferability between different densities, and the learned potential also agrees well with the ground-truth LJ potential. Identically to the DiffSim result, the IBI result shows excellent agreement with the ground-truth potential at low density and also generalizes to higher densities. As the density of the system used for fitting increases, the learned potentials show high variability between replicate experiments, with overall poorer transferability to systems that are not used for training. Many of the learned potentials are dissimilar to the ground-truth potential, featuring concave-down shapes. For a high-density system with $\rho = 0.7$, neither IBI nor DiffSim produces potentials that resemble the ground-truth LJ potential. The learned potentials for $\rho = 0.7$ also do not transfer well to $\rho = 0.3, 0.5$ with varied magnitudes of deviation. Interestingly, IBI tends to yield more shallow fits at higher density whereas DiffSim over pronounces the features of the LJ potential with a deeper minimum and a higher maximum.

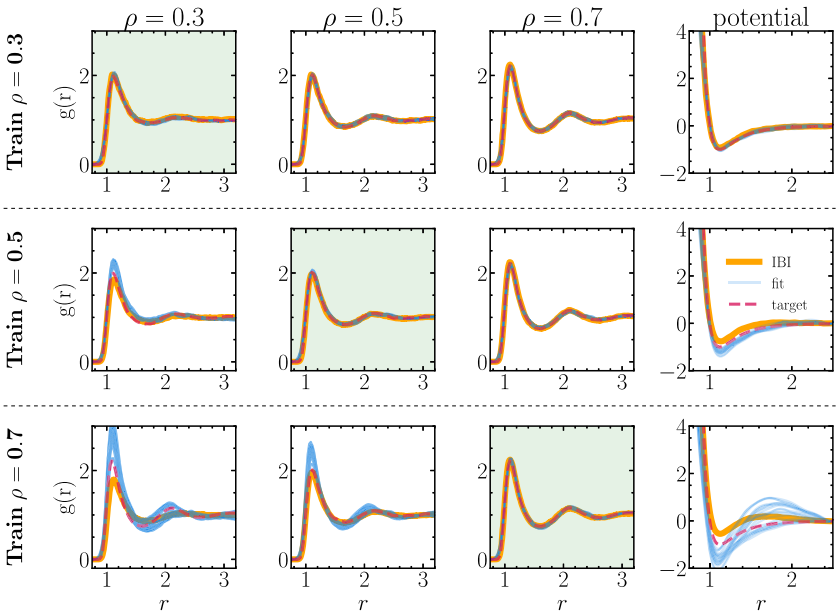

FIG. 2. Transferability of DiffSim and IBI potentials fitted to reproduce $g(r)$ of Lennard-Jones potentials at different densities. Each row contains the results for the simulated $g(r)$ at three densities (0.3, 0.5, and 0.7 from left to right) with potential fitted at a particular density (light green background, 0.3, 0.5, and 0.7 top to bottom) The last column reports the learned potentials compared the ground truth. We include learned pair potentials obtained from ten independent runs.

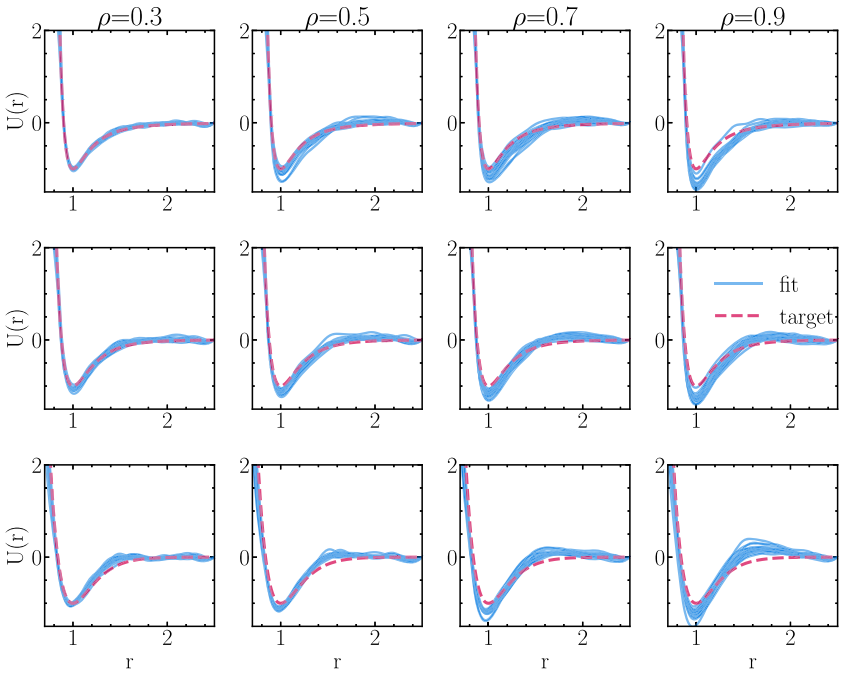

FIG. 3. Comparison of fitted potentials with modified Morse potentials MM(6.5, −0.45) (top), MM(5.5, 0.44) (middle), and MM(4.5, 1.52) (bottom). The figure includes learned pair potentials obtained from ten independent runs.

To demonstrate that the observed trend is not specific to the LJ potential, we tested our methods on the Modified Morse (MM) Potentials with different softness at different densities.[67] The MM potential takes the following form,

$$MM(\rho,\psi)(r) = \frac{e^{2\rho(1-r^{\psi})} - 2e^{\rho(1-r^{\psi}} - A)}{1+A} \tag{14}$$

where $A = 0$ for $\psi \geq 0$ and $A = e^{2\rho/\psi} - 2e^{\rho/\psi}$ for $\psi < 0$. One can tune (ρ, ψ) to generate a diverse set of pair potentials. We tested our method for three different potentials MM(6.5, −0.45), MM(5.5, 0.44), and MM(4.5, 1.52) at four different densities $\rho = (0.3, 0.5, 0.7, 0.9)$ to infer pair potentials from target RDFs in those systems. For quantitative evaluation, we calculate the average deviation of learned potentials from the ground truth using the metric:

$$\Delta = \int_{r_0}^{r_c} |U_{fit}(r) - U_{true}(r)| dr \tag{15}$$

where we choose $r_c = 2.5$ and $r_0 = 0.9$. Figures 3 and 4 show that the learned potentials deviate less from the ground truth as the density decreases, which is consistent with the trend observed for LJ systems. This is also in line with observations in the past literature.[32] For denser systems, we attribute the low sensitivities of the learned potentials to dense local packing that downplays the effect of intermolecular forces, largely due to the "entropic effect of packing."[68,69] Another factor is the lowered diffusion in denser systems, and thus less configuration space is explored over the same simulation time span.

B. Learning pair potentials for binary systems

Our DiffSim-based learning protocol is highly flexible. For example, it can be modified to incorporate multiple learning targets to learn pair interactions between different particle types. During training, this involves the simultaneous optimization of different interaction functions between different types of particles.

To demonstrate the efficacy of our method in this case, we construct a similar example as presented in Rosenberger and van der Vegt.[70] The binary mixture system involves three types of LJ interactions with the following parameters: $\{(\varepsilon_{AA} : 1.0, \sigma_{AA} : 0.9), (\varepsilon_{AB} : 1.0, \sigma_{AB} : 1.0), (\varepsilon_{BB} : 1.0, \sigma_{BB} : 1.1)\}$. We simulate 500 000 steps at a temperature of $k_BT = 1.0$ with a time step of 0.005 to generate the target RDFs for $x = (0.25, 0.5, 0.75)$, where x is the mole fraction of particle A: $\frac{n_A}{n_A+n_B}$. We optimize based on targets at $x = 0.25$ and test the transferability of the learned potential at

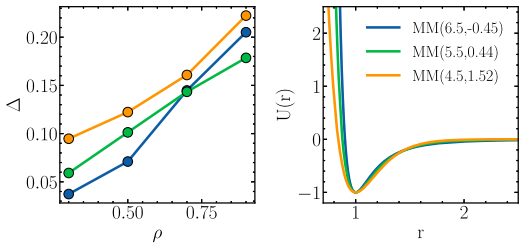

FIG. 4. Left: Calculated mean deviation (Δ) between the fitted potentials and the ground-truth modified Morse potentials. For the three potentials tested, the learned potentials manifest less deviation as density decreases. Right: Modified Morse potentials used for learning.

$x = (0.5, 0.75)$. During the simulation, we learn the three types of interaction simultaneously by optimizing a joint loss of equally weighted loss terms.

$$\begin{aligned} L &= L_{AA} + L_{AB} + L_{BB} \\ &= \int_r (g^{AA}_{\text{sim.}}(r) - g^{AA}_{\text{ref.}}(r))^2 dr + \int_r (g^{AB}_{\text{sim.}}(r) - g^{AB}_{\text{ref.}}(r))^2 dr \\ &\quad + \int_r (g^{BB}_{\text{sim.}}(r) - g^{BB}_{\text{ref.}}(r))^2 dr \end{aligned} \tag{16}$$

In Fig. 5, we show that DiffSim accurately reproduces the three target RDFs for $x = 0.25$. However, as the learned potential correctly recovers the repulsive part of the target potential, the attractive part shapes are dissimilar to the ground truth and feature a concave-down shape. As a transferability test, we apply the potential learned at $x = 0.25$ to systems with different compositions, that is $x = 0.5$ and $x = 0.75$, and observe poor transferability. Specifically, the learned potentials overestimate the height of the first peak for the target RDFs at $x = 0.5$ and $x = 0.75$. In this case, the learned potential overfits to the composition state point on which it was trained.

For improved transferability, our model can also be configured to train systems simultaneously in different states. This involves applying DiffSim to simulate multiple systems and combining multiple gradient updates to the potential arising from the different trajectories. In practice, we initialize all three systems and simulate them in parallel with the same shared potentials. Gradient updates are collected through Eqs. (2) and (9) from each trajectory, and the pair potentials of AA, AB, and BB are updated with the combined three gradients from the three state points. For each training epoch, three systems are simulated with gradients accumulated on the parameters of the pair potential with the total loss as the sum of individual losses evaluated for each system, i.e., $L_{tot} = L_{x=0.25} + L_{x=0.5} + L_{x=0.75}$. Each loss takes the form described in Eq. (16), giving us nine optimization targets in total, namely the RDF of AA, AB, and BB at the three concentrations. Our training results in Fig. 6 show that potentials can be learned to simultaneously fit target RDFs at multiple systems and state points at the same time. Compared with potentials solely fitted at $x = 0.25$ (Fig. 5), the learned potentials in all three states show a much better resemblance to ground-truth LJ potentials for both repulsive and attractive portions. The learned potentials are thus able to reproduce the structural properties of binary mixtures with mole fractions that are not included in the training.

C. Learning transferable coarse-grained potentials

As the third example, we show that DiffSim can learn pair potentials for CG simulations of water with a temperature-transferable functional form. For the ground truth simulations, we simulate all-atom water molecules at 1 g/cm^3 at 288, 338, and 388 K with the SPC/E force field.[10] We choose to map each water molecule onto the oxygen atom and use the oxygen–oxygen RDF as our fitting target. We first run a single-state fitting experiment at single temperatures and use the learned potentials to simulate systems at

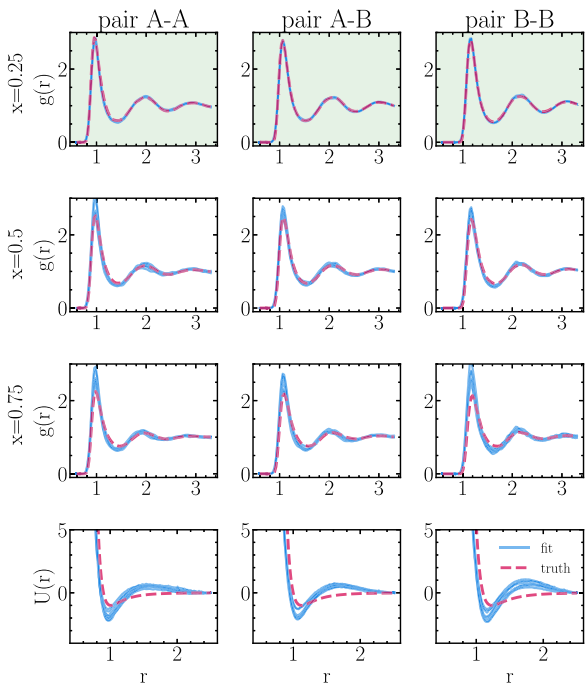

FIG. 5. We apply the pair potentials learned at $x = 0.25$ (highlighted with light green background) to systems with $x = (0.5, 0.75)$. Unlike ground truth potentials of the LJ type, the learned potentials feature concave-down shapes.

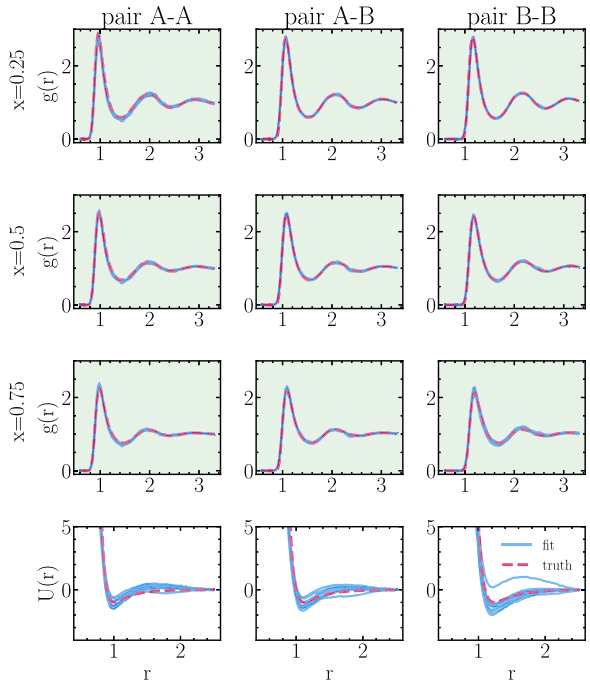

FIG. 6. RDFs and pair potentials obtained from simultaneous learning of pair potentials for systems at $x = 0.25, x = 0.5, x = 0.75$. The learned potentials resemble the ground truth potentials. The learned potentials accurately recover the RDFs at all compositions.

other temperatures. For comparison, we also performed the IBI at all three temperatures. Figure 7 shows the fitting and transferability results. As DiffSim and IBI manage to accurately fit single states, they manifest large RDF errors when using similar systems at other temperatures. The learned potentials using IBI and DiffSim show similar shapes, with an inner potential well at 2.7 Å, approximately where the RDF of water has the first peak. Furthermore, the depth decreases with temperature, showing a clear temperature dependence of the pair potential. This observation agrees with the water potentials obtained in the past literature,[71,72] suggesting that an explicit temperature dependence is required for the pair potential to be transferable.

To incorporate the temperature-dependent nature of CG potentials, we made our neural pair potentials temperature-dependent for improved transferability across temperatures. With AD, our method enables more flexible forms of pair potentials than single- or multi-state IBI. Specifically, we optimize pair potentials with explicit temperature dependence. Inspired by previous work studying the energy and entropy decomposition of CG potentials,[72,73] we propose the following form of neural pair potential,

$$u_T(r, T) = u_1(r) - k_B T u_2(r), \tag{17}$$

where u_1 and u_2 are two distinct neural pair potentials. $u_1(r)$ can be understood as the energy contribution to pair potentials, and $u_2(r)$ is the entropy contribution. For training, we initialize three coarse-grained water systems and simulate them at different temperatures. We simulate and train the three systems simultaneously to optimize $u_1(r)$ and $u_2(r)$ jointly. In Fig. 8, we show the RDFs obtained and the learned pair potentials at the three different temperatures used for learning. The learned potentials at different temperatures show similar shapes. All three potentials have a characteristic barrier of around 3.2 Å, and the change in temperature affects their height. Compared with the potential obtained by fitting on single states, the learned temperature-dependent pair potential shows excellent transferability to all three temperatures used for optimization. To quantify the deviation between the simulated RDF and the target RDF, we use the following metric proposed by Pretti *et al.*,

$$D = 4\pi\rho \int_0^{r_c} r^2 (g_{(r)} - g_{\text{ref.}}(r))^2 dr. \tag{18}$$

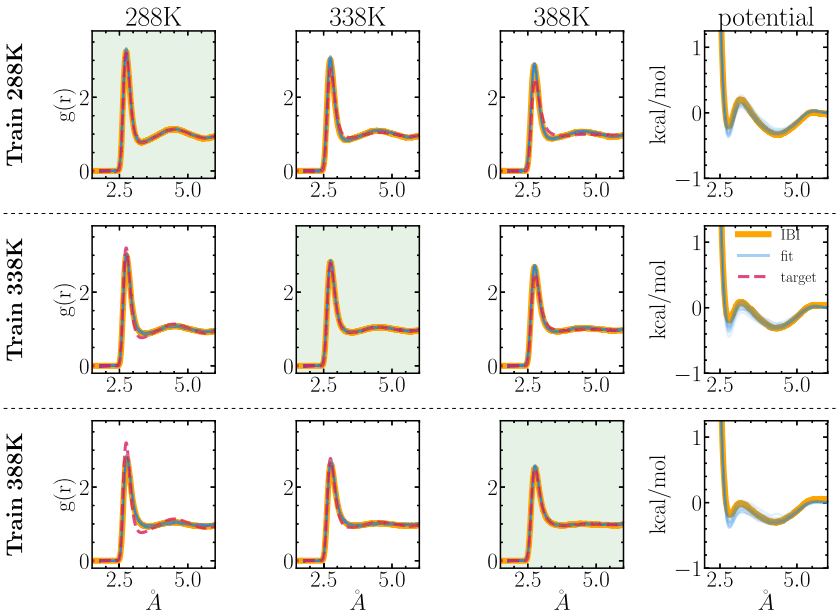

FIG. 7. Learning coarse-grained water potential at single state points and transfer to other state points. The results include learned pair potentials and simulated RDFs from 20 independent runs with DiffSim. The learned potentials using IBI are also included.

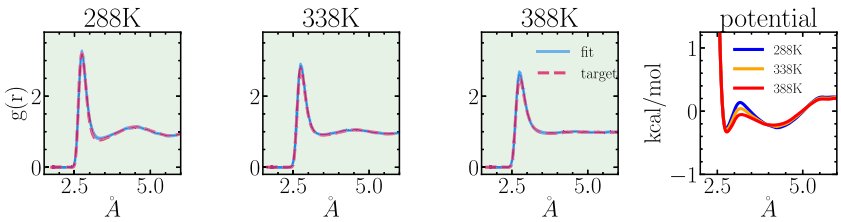

FIG. 8. Results for the leaned temperature-dependent pair potential. Simulated RDFs and learned transferable potential at 288, 338, and 388 K. The learned potential shows a temperature-dependent barrier height, which increases with decreasing temperature.

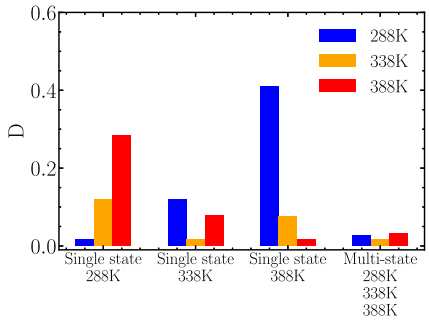

FIG. 9. Quantified RDF error (D) of learned potentials simulated at different state points (288, 338, and 388 K).

In Fig. 9, we compare the deviation of the simulated RDF obtained from different training protocols. The result shows that fitting in a single state can produce an accurate RDF in the state used for fitting, but it does not transfer well to other states. The temperature-transferable potential produces overall low RDF deviations for all three states, only slightly larger than those of the model optimized at the specific temperature.

IV. CONCLUSION

In this work, we propose a flexible method based on DiffSim to directly learn pair potentials. We demonstrate our methods in several computational examples. First, we apply our method to learning simple pair potentials, where the ground truth is known. We show that our method recovers a diverse set of possible pair potentials given a target RDF and, therefore, can be used to probe sensitivities for fitting pair potentials. For a dense system, our results are in line with previous reports, indicating that fitted potentials can be highly insensitive.[64,69] In contrast, fitting the dilute system shows a better sensitivity, with the obtained potentials in reasonable agreement with the ground truth potential. We further demonstrate the multistate fitting capabilities of the method by applying it to binary mixtures, which involves learning multiple pair interactions under multiple composition conditions simultaneously. Learning involves the incorporation of multiple loss functions for different RDF targets. Our experiments show that simultaneous learning on systems with different compositions is possible and produces potentials that have better transferability. Additionally, we apply our method to learning CG pair potentials. We demonstrate how learning a single temperature-dependent CG potential for water at three different temperatures is more transferable than any potential obtained from fitting only a single state point.

The results show clearly that a naive (unoptimized with respect to hyperparameters) application of DiffSim achieves qualitatively identical results to those obtained by IBI, while using significantly fewer MD time steps. Additionally, unlike IBI, DiffSim can be directly applied to optimizing several objectives and learning temperature-dependent potentials as well. Therefore, the proposed DiffSim protocol can be understood as a gradient-based extension of the iterative Boltzmann inversion. Our method directly optimizes potentials with the gradient signal obtained from simulations. It allows simultaneous simulation and fitting for systems at different temperatures and compositions, allowing direct parameterization of pair potentials to improve transferability. The learning pipeline can be flexibly set up with the incorporation of multiple optimization objectives and allows for the optimization of a broader range of pair potentials, such as pair potentials with explicit thermodynamic dependence. We anticipate that the proposed framework provides improved modeling flexibility for multi-scale simulation of molecular liquids.

DiffSim learns potentials by directly optimizing unrolled simulation operations that involve updates of positions, velocities, and forces. The procedure directly learns to match an observation without atomistic data, which requires extensive sampling that often requires active learning. Compared with force-based optimization of machine learned potentials, our method produces direct simulation feedback and guarantees the stable production of desired observables. When only relying on matching atomistic forces, highly flexible neural force fields often suffer from unstable dynamics, which is common even for models showing very low force errors.[25] In comparison, pair potentials constrain interactions between pairs of particles and are, therefore, easier to interpret and more efficient to simulate than complex neural force fields. Although observable-informed and gradient-based, our learning protocol can also be combined with force matching to incorporate top-down and bottom-up parametrization into optimization.

Our method opens up broader possibilities for learning-based simulations: objectives can be extended to include more observables, such as stress tensors and velocity autocorrelation functions, to enable gradient-based top-down parameterization of force fields. However, many interesting targets require either extensive sampling or long continuous trajectories. At this stage, DiffSim is limited to short time scales as gradients become unstable after many time steps. Ensemble averages of coordinate dependent observables are accessible through differentiable reweighting, as proposed in Ref. 45, whereas kinetic properties like autocorrelation functions can only be optimized through DiffSim. Future work is needed to afford more and especially longer sampling per DiffSim epoch.

ACKNOWLEDGMENTS

This work was supported by Toyota Research Institute. J.C.B.D. is thankful for the support of the Leopoldina Fellowship Program, German National Academy of Sciences Leopoldina, Grant No. LPDS 2021-08.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

W.W. and Z.W. contributed equally to this work.

Wujie Wang: Conceptualization (equal); Investigation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Zhenghao Wu**: Conceptualization (supporting); Investigation (equal); Writing – original draft (equal); Writing – review &

editing (equal). **Johannes C. B. Dietschreit**: Writing – review & editing (equal). **Rafael Gómez-Bombarelli**: Conceptualization (equal); Funding acquisition (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are openly available in https://github.com/torchmd/mdgrad.

APPENDIX: COMPUTATIONAL DETAILS

1. Model and training hyper-parameters

Here, we detail the hyper-parameters used in our experiments. For all training runs, a learning rate scheduler is used to dynamically decrease the learning when the pre-defined convergence criterion is met. The learning rate is expected to decrease by half when the improvement in training loss is less than 0.1%.

Lennard-Jones and MM systems. The LJ and MM systems are simulated at the densities $\rho = (0.3, 0.5, 0.7, 0.9)$ with cubic box edge lengths $L = (9.49, 8.00, 7.15, 6.58)$, respectively. The model is trained for 500 epochs, with each epoch comprising 120 simulation steps. The learning rate is set at 0.002. Neural pair potentials are constructed with Gaussian-smeared distance inputs of 100 bins $(K = 100)$ with $d = 0.1$. The smeared inputs are fed into a multilayer perceptron of three hidden layers with a hidden layer size of 128. The activation function that we use is exponential linear unit (ELU).[74] The prior potential $u_{prior}(r)$ that we use takes the form $\varepsilon(\frac{\sigma}{r})^{10}$ with $\sigma = 0.9$ and $\varepsilon = 0.4$. During simulations, the neighbor list is updated for every integration step.

Binary mixtures. The binary mixture systems are simulated at three mole fractions, $x = (0.25, 0.5, 0.75)$, at a fixed density of $\rho = 0.8$ with a cubic box edge length $L = 6.84$. For single-state learning (mole fraction $x = 0.25$), we optimize three potentials for 300 epochs using the Adam optimizer with a learning rate of 0.001, with each epoch consisting of 250 MD steps. For multi-state learning (mole fractions $x = 0.25, 0.5, 0.75$), the model is trained for 500 epochs, with each epoch comprising 120 simulation steps. The learning rate is set at 0.0003. Three neural pair potentials are initialized to represent the three types of pair interaction involved in the system. For the computation of the RDF loss, each pair of statistics is equally weighted. Each pair potential uses 100 bins $(K = 100)$ to smear all distance inputs with $d = 0.15$. The MLP consists of three hidden layers with 128 neurons each. The activation function used is scaled exponential linear unit (SELU).[75] The prior potential $u_{prior}(r)$ we use the form $\varepsilon(\frac{\sigma}{r})^6$ with $\sigma = 1.0$ and $\varepsilon = 2.0$. Before training, a pre-training step is performed to initialize neural network potentials with shapes similar to the Boltzmann inverted potentials derived from the target RDF. This step is used to facilitate faster convergence of training and sampling of relevant parts of the configuration space.

Temperature-transferable water simulations. The model is trained for 500 epochs, with each epoch comprising 190 simulation steps of 2 fs each. The learning rate is set at 0.000 65. We similarly use three-layer neural networks to represent u_1 and u_2 with hidden layers of 115 neurons. The pair distances are smeared with $K = 120$ equally spaced Gaussian kernels with $d = 0.15$. The activation function used is ELU.[74] u_{prior} is used with $\sigma = 40$ kcal/mol and $\varepsilon = 1.7$ Å. A pre-training procedure is also performed using the Boltzmann inverted RDF at 288 K as the target.

2. Supplementary results

We provide additional results in Fig. 10 to show the RDFs generated by learned pair potentials for different MM systems at different densities. Figure 10 provides supplementary information for Fig. 3. The simulations are carried out for 100 000 steps with each step being 0.005.

3. Molecular dynamics simulations of water

The FG model of water used in this work is an all-atom water model described by SPC/E force field.[10] The large-scale atomic/molecular massively parallel simulator (LAMMPS)[76,77] is used to perform MD simulations of systems with $N = 1500$ water molecules at constant density $\rho = 1.0$ g/cm^3 and at temperatures of $T = 288K$, $T = 338K$ and $T = 388K$, with a time unit $\delta t = 1$ fs. We apply periodic boundary conditions in the x, y, and z directions, and the Nosé–Hoover thermostat[78,79] with a damping time of $\tau = 100$ fs. The water systems are first equilibrated for 10 ns. Subsequently, production runs of 1 ns are carried out to collect data every 0.1 ps to calculate the reference radial distribution functions of oxygen atoms with bin size 0.001 nm. In the CG representation, a water molecule is replaced by a single bead

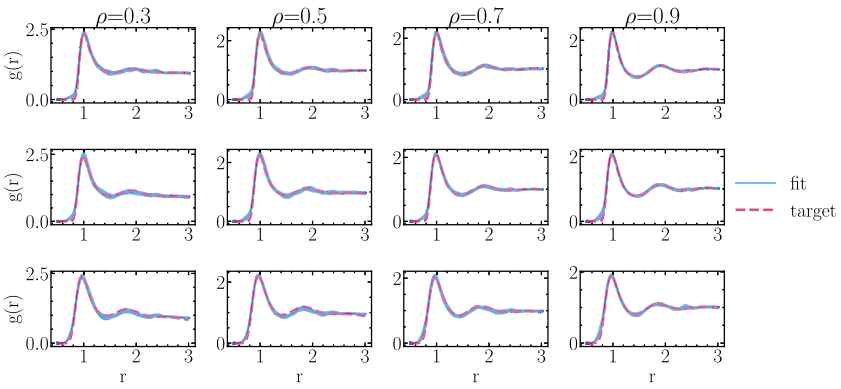

FIG. 10. Comparison between ground-truth RDFs and RDFs from simulations using learned pair potentials for different MM potentials at different densities. From top to bottom rows, the figures correspond to RDFs generated by learned potentials for MM(6.5, −0.45), MM(5.5, 0.44), and MM(4.5, 1.52) systems, respectively.

placed on the oxygen atom. CG simulations are also performed in LAMMPS.[76,77]

4. Protocol for iterative Boltzmann inversion

Lennard-Jones systems. The RDFs of the MD simulations with LJ potentials are the target distribution fed to the IBI machinery. The cutoff radius used for IBI optimization and simulations with IBI refined potential is $r_{cut} = 2.0$ with space size $\delta r = 0.01\sigma$. For each state point, the MD simulation performed under the canonical ensemble in each IBI iteration consisted of 5×10^5 equilibration steps and 5×10^5 sampling steps for the calculation of the RDF. The entire IBI optimization takes between 10 and 20 iterations to yield converged distributions that match the target RDFs.

Water simulations. The RDFs from the SPC/E water simulation are obtained by mapping the CG beads mapped to the oxygen atoms and are fed into the IBI optimization protocol. The cutoff radii used for IBI optimization and CG simulations are $r_{cut} = 0.625$ nm with a space size $\delta r = 0.0005$ nm. For each state point, the CG-MD simulation performed in the canonical ensemble in each IBI iteration consisted of 2×10^6 equilibration steps and 1×10^6 sampling steps for the calculation of the RDF. The entire IBI optimization takes between 10 and 20 iterations to yield converged distributions that match the target RDFs.

REFERENCES

[1] D. Vlachakis, E. Bencurova, N. Papangelopoulos, and S. Kossida, "Current state-of-the-art molecular dynamics methods and applications," Adv. Protein Chem. Struct. Biol. **94**, 269–313 (2014).

[2] J. Blumberger, "Recent advances in the theory and molecular simulation of biological electron transfer reactions," Chem. Rev. **115**, 11191–11238 (2015).

[3] T. E. Gartner and A. Jayaraman, "Modeling and simulations of polymers: A roadmap," Macromolecules **52**, 755–786 (2019).

[4] J. Fish, G. J. Wagner, and S. Keten, "Mesoscopic and multiscale modelling in materials," Nat. Mater. **20**, 774–786 (2021).

[5] S. L. Mayo, B. D. Olafson, and W. A. Goddard, "DREIDING: A generic force field for molecular simulations," J. Phys. Chem. **94**, 8897–8909 (1990).

[6] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, "Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids," J. Am. Chem. Soc. **118**, 11225–11236 (1996).

[7] R. W. Pastor and A. D. MacKerell, "Development of the CHARMM force field for lipids," J. Phys. Chem. Lett. **2**, 1526–1532 (2011).

[8] D. C. Rapaport, *The Art of Molecular Dynamics Simulation* (Cambridge University Press, Cambridge, New York, UK, 2004), oCLC: 928698870.

[9] G. A. Cisneros, K. T. Wikfeldt, L. Ojamäe, J. Lu, Y. Xu, H. Torraifard, A. P. Bartók, G. Csányi, V. Molinero, and F. Paesani, "Modeling molecular interactions in water: From pairwise to many-body potential energy functions," Chem. Rev. **116**, 7501–7528 (2016).

[10] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma, "The missing term in effective pair potentials," J. Phys. Chem. **91**, 6269–6271 (1987).

[11] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, "Comparison of simple potential functions for simulating liquid water," J. Chem. Phys. **79**, 926–935 (1983).

[12] J. Glaser, T. D. Nguyen, J. A. Anderson, P. Lui, F. Spiga, J. A. Millan, D. C. Morse, and S. C. Glotzer, "Strong scaling of general-purpose molecular dynamics simulations on GPUs," Comput. Phys. Commun. **192**, 97–107 (2015).

[13] A. Albaugh, H. A. Boateng, R. T. Bradshaw, O. N. Demerdash, J. Dziedzic, Y. Mao, D. T. Margul, J. Swails, Q. Zeng, D. A. Case, P. Eastman, L.-P. Wang, J. W. Essex, M. Head-Gordon, V. S. Pande, J. W. Ponder, Y. Shao, C.-K. Skylaris, I. T. Todorov, M. E. Tuckerman, and T. Head-Gordon, "Advanced potential energy surfaces for molecular simulation," J. Phys. Chem. B **120**, 9811–9832 (2016).

[14] J. Wang, N. Charron, B. Husic, S. Olsson, F. Noé, and C. Clementi, "Multi-body effects in a coarse-grained protein force field," J. Chem. Phys. **154**, 164113 (2021).

[15] S. J. Plimpton and A. P. Thompson, "Computational aspects of many-body potentials," MRS Bull. **37**, 513–521 (2012).

[16] F. Müller-Plathe, "Coarse-graining in polymer simulation: From the atomistic to the mesoscopic scale and back," ChemPhysChem **3**, 754–769 (2002).

[17] P. C. T. Souza, R. Alessandri, J. Barnoud, S. Thallmair, I. Faustino, F. Grünewald, I. Patmanidis, H. Abdizadeh, B. M. H. Bruininks, T. A. Wassenaar, P. C. Kroon, J. Melcr, V. Nieto, V. Corradi, H. M. Khan, J. Domański, M. Javanainen, H. Martinez-Seara, N. Reuter, R. B. Best, I. Vattulainen, L. Monticelli, X. Periole, D. P. Tieleman, A. H. de Vries, and S. J. Marrink, "Martini 3: A general purpose force field for coarse-grained molecular dynamics," Nat. Methods **18**, 382–388 (2021).

[18] S. Dhamankar and M. A. Webb, "Chemically specific coarse-graining of polymers: Methods and prospects," J. Polym. Sci. **59**, 2613–2643 (2021).

[19] F. Ercolessi and J. B. Adams, "Interatomic potentials from first-principles calculations: The force-matching method," Europhys. Lett. **26**, 583–588 (1994).

[20] S. Izvekov and G. A. Voth, "A multiscale coarse-graining method for biomolecular systems," J. Phys. Chem. B **109**, 2469–2473 (2005).

[21] W. G. Noid, J.-W. Chu, G. S. Ayton, and G. A. Voth, "Multiscale coarse-graining and structural correlations: Connections to liquid-state theory," J. Phys. Chem. B **111**, 4116–4127 (2007).

[22] L. Zhang, J. Han, H. Wang, R. Car, and W. E, "DeePCG: Constructing coarse-grained models via deep neural networks," J. Chem. Phys. **149**, 034101 (2018).

[23] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. de Fabritiis, F. Noé, and C. Clementi, "Machine learning of coarse-grained molecular dynamics force fields," ACS Cent. Sci. **5**, 755–767 (2019).

[24] J. F. Rudzinski and W. G. Noid, "The role of many-body correlations in determining potentials for coarse-grained models of equilibrium structure," J. Phys. Chem. B **116**, 8621–8635 (2012).

[25] S. Stocker, J. Gasteiger, F. Becker, S. Günnemann, and J. T. Margraf, "How robust are modern graph neural network potentials in long and hot molecular dynamics simulations?," Mach. Learn.: Sci. Technol. **3**, 045010 (2022).

[26] A. Chaimovich and M. S. Shell, "Coarse-graining errors and numerical optimization using a relative entropy framework," J. Chem. Phys. **134**, 094112 (2011).

[27] E. Kocer, T. W. Ko, and J. Behler, "Neural network potentials: A concise overview of methods," Annu. Rev. Phys. Chem. **73**, 163–186 (2022).

[28] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations: Mesoscale potentials from atomistic simulations," J. Comput. Chem. **24**, 1624–1636 (2003).

[29] H.-J. Qian, P. Carbone, X. Chen, H. A. Karimi-Varzaneh, C. C. Liew, and F. Müller-Plathe, "Temperature-transferable coarse-grained potentials for ethylbenzene, polystyrene, and their mixtures," Macromolecules **41**, 9919–9929 (2008).

[30] H. A. Karimi-Varzaneh, F. Müller-Plathe, S. Balasubramanian, and P. Carbone, "Studying long-time dynamics of imidazolium-based ionic liquids with a systematically coarse-grained model," Phys. Chem. Chem. Phys. **12**, 4714 (2010).

[31] J. W. Mullinax and W. G. Noid, "Extended ensemble approach for deriving transferable coarse-grained potentials," J. Chem. Phys. **131**, 104110 (2009).

[32] T. C. Moore, C. R. Iacovella, and C. McCabe, "Derivation of coarse-grained potentials via multistate iterative Boltzmann inversion," J. Chem. Phys. **140**, 224104 (2014).

[33] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, "Differentiable programming tensor networks," Phys. Rev. X. **9**, 031041 (2019).

[34] C. Schenck and D. Fox, "SPNets: Differentiable fluid dynamics for deep neural networks," in *Conference on Robot Learning* (PMLR, 2018), pp. 317–335.

[35] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Machine learning force fields," Chem. Rev. **121**, 10142–10186 (2021).

[36]M. F. Kasim and S. M. Vinko, "Learning the exchange-correlation functional from nature with fully differentiable density functional theory," Phys. Rev. Lett. **127**, 126403 (2021).
[37]B. Kanungo, P. M. Zimmerman, and V. Gavini, "Exact exchange-correlation potentials from ground-state electron densities," Nat. Commun. **10**, 4497 (2019).
[38]T. Tamayo-Mendoza, C. Kreisbeck, R. Lindh, and A. Aspuru-Guzik, "Automatic differentiation in quantum chemistry with applications to fully variational Hartree–Fock," ACS Cent. Sci. **4**, 559–566 (2018).
[39]M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, "Efficient and modular implicit differentiation," arXiv:2105.15183 (2021).
[40]J. R. Magnus, "On differentiating eigenvalues and eigenvectors," Econometric Theory **1**, 179–191 (1985).
[41]R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems* (NeurIPS, 2018), Vol. 31.
[42]W. Wang, S. Axelrod, and R. Gómez-Bombarelli, "Differentiable molecular simulations for control and learning," arXiv:2003.00868 (2020).
[43]S. Schoenholz and E. D. Cubuk, "JAX M.D.: A framework for differentiable physics," in *Advances in Neural Information Processing Systems* (NeurIPS, 2020), Vol. 33, pp. 11428–11441.
[44]S. Doerr, M. Majewski, A. Pérez, A. Krämer, C. Clementi, F. Noe, T. Giorgino, and G. De Fabritiis, "TorchMD: A deep learning framework for molecular simulations," J. Chem. Theory Comput. **17**, 2355–2363 (2021).
[45]S. Thaler and J. Zavadlav, "Learning neural network potentials from experimental data via differentiable trajectory reweighting," Nat. Commun. **12**, 6884 (2021).
[46]J. G. Greener and D. T. Jones, "Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins," PLoS One **16**, e0256990 (2021).
[47]A. Moradzadeh and N. R. Aluru, "Transfer-learning-based coarse-graining method for simple fluids: Toward deep inverse liquid-state theory," J. Phys. Chem. Lett. **10**, 1242–1250 (2019).
[48]P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," arXiv:1806.01261 (2018).
[49]J. Baxter, "A model of inductive bias learning," J. Artif. Intell. Res. **12**, 149–198 (2000).
[50]J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," Proc. Natl. Acad. Sci. U. S. A. **115**, 8505–8510 (2018).
[51]Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," Biostat. Epidemiol. **1**, 161–187 (2017).
[52]S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2022).
[53]M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, 2016), pp. 265–283.
[54]R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," in *Systems for Machine Learning* (SysML, 2018), pp. 23–24.
[55]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (NeurIPS, 2019), Vol. 32.
[56]E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss," in *Advances in Neural Information Processing Systems* (NeurIPS, 2016), Vol. 29.
[57]K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks **2**, 359–366 (1989).
[58]W. Tschöp, K. Kremer, J. Batoulis, T. Bürger, and O. Hahn, "Simulation of polymer melts. I. Coarse-graining procedure for polycarbonates," Acta Polym. **49**, 61–74 (1998).
[59]N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," arXiv:1810.02513 (2018).
[60]G. J. Martyna, M. L. Klein, and M. Tuckerman, "Nosé–Hoover chains: The canonical ensemble via continuous dynamics," J. Chem. Phys. **97**, 2635–2643 (1992).
[61]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[62]J. Ingraham, A. Riesselman, C. Sander, and D. Marks, "Learning protein structure with a differentiable simulator," in International Conference on Learning Representations, 2018.
[63]L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," arXiv:2111.05803 (2021).
[64]R. Potestio, "Is Henderson's theorem practically useful?," JUnQ **3**, 13–15 (2013).
[65]D. Rosenberger, M. Hanke, and N. F. A. van der Vegt, "Comparison of iterative inverse coarse-graining methods," Eur. Phys. J.: Spec. Top. **225**, 1323–1345 (2016).
[66]V. Rühle, C. Junghans, A. Lukyanov, K. Kremer, and D. Andrienko, "Versatile object-oriented toolkit for coarse-graining applications," J. Chem. Theory Comput. **5**, 3211–3223 (2009).
[67]L. Cheng and J. Yang, "Modified Morse potential for unification of the pair interactions," J. Chem. Phys. **127**, 124104 (2007).
[68]D. Chandler, J. D. Weeks, and H. C. Andersen, "Van der Waals picture of liquids, solids, and phase transformations," Science **220**, 787–794 (1983).
[69]H. Wang, F. H. Stillinger, and S. Torquato, "Sensitivity of pair statistics on pair potentials in many-body systems," J. Chem. Phys. **153**, 124106 (2020).
[70]D. Rosenberger and N. F. A. van der Vegt, "Relative entropy indicates an ideal concentration for structure-based coarse graining of binary mixtures," Phys. Rev. E **99**, 053308 (2019).
[71]M. E. Johnson, T. Head-Gordon, and A. A. Louis, "Representability problems for coarse-grained water potentials," J. Chem. Phys. **126**, 144509 (2007).
[72]E. Pretti and M. S. Shell, "A microcanonical approach to temperature-transferable coarse-grained models using the relative entropy," J. Chem. Phys. **155**, 094102 (2021).
[73]K. M. Kidder, R. J. Szukalo, and W. Noid, "Energetic and entropic considerations for coarse-graining," Eur. Phys. J. B **94**, 153 (2021).
[74]D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," arXiv:1511.07289 (2015).
[75]G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems* (NeurIPS, 2017), Vol. 30.
[76]S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," J. Comput. Phys. **117**, 1–19 (1995).
[77]A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen *et al.*, "LAMMPS—A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," Comput. Phys. Commun. **271**, 108171 (2022).
[78]S. Nosé, "A unified formulation of the constant temperature molecular dynamics methods," J. Chem. Phys. **81**, 511–519 (1984).
[79]W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions," Phys. Rev. A **31**, 1695–1697 (1985).