A GPU-Accelerated Machine Learning Framework for Molecular Simulation: HOOMD-blue with TensorFlow

R. Barrett,¹ M. Chakraborty,¹ D. Amirkulova,¹ H. Gandhi,¹ and A. White^{1, a)}

Chemical Engineering, University of Rochester.

As interest grows in applying machine learning force-fields and methods to molecular simulation, there is a need for state-of-the-art inference methods to use trained models within efficient molecular simulation engines. We have designed and implemented software that enables integration of a scalable GPU-accelerated molecular mechanics engine, HOOMD-blue, with the machine learning (ML) TensorFlow package. TensorFlow is a GPU-accelerated, scalable, graph-based tensor computation model building package that has been the implementation of many recent innovations in deep learning and other ML tasks. TensorFlow models are constructed in Python and can be visualized or debugged using the rich set of tools implemented in the TensorFlow package. In this article, we present four major examples of tasks this software can accomplish which would normally require multiple different tools: (1) we train a neural network to reproduce a force field of a Lennard-Jones simulation; (2) we perform online force matching of methanol; (3) we compute the maximum entropy bias of a Lennard-Jones collective variable; (4) we calculate the scattering profile of an ongoing TIP4P water molecular dynamics simulation. This work should accelerate both the design of new neural network based models in computational chemistry research and reproducible model specification by leveraging a widely-used ML package

I. INTRODUCTION

HOOMD-blue^{1,2} is a GPU-accelerated engine for molecular dynamics (MD) and hard particle Monte Carlo simulations, and has simulated clathrate crystal colloids,³ coarse grained solar cell polymers,⁴ intrinsically disordered proteins,⁵ and various other systems.^{6–24} HOOMD-blue can simulate large systems with high speed due to the scalable nature of GPU processing. Another advantage of HOOMD-blue is that it has a Python interface that can be readily incorporated into a larger Python workflow.

TensorFlow²⁵ is a ML library created and maintained by Google. TensorFlow uses a graph-based computation framework with tensor operations to represent its underlying mathematical operations. This tensor graph abstraction allows for flexible model design, where one can easily add, remove, or alter operations (nodes) of a model (graph) while preserving the overall structure and flow of the tensor data (edges). TensorFlow has a built-in visualization tool called TensorBoard²⁵ to aid in this process. One consequence of this tensor computation graph model designation is a major advantage of TensorFlow: most of the tensor operations defined in its library are analytically differentiable, and these derivatives can be automatically propagated throughout any TensorFlow model upon request. Thus, as long as we can express a model as a composition of tensor operations, we automatically have access to its derivatives at every step to compute forces or perform learning. Additionally, TensorFlow can optimize model evaluation by caching values of tensors as they are evaluated so that branches in the graph need not have these values explicitly stored in memory.

The HTF package described in this work gives TensorFlow access to the per-particle positions, neighbor lists, and forces generated by HOOMD-blue at each time step of a simulation. Since both HOOMD-blue and TensorFlow can execute

entirely on the GPU, there is minimal loss of speed due to communication. With HTF, a user may specify any tensor operation on the neighbor list and automatically calculate its derivatives and thus, its forces. This enables use of arbitrary force fields (e.g., from neural networks), calculation of arbitrary collective variables, and biasing of arbitrary collective variables. In addition, since TensorFlow contains a suite of ML algorithm implementations, a user can also perform learning on any of the items calculated using HTF.

ML has been defined a number of different ways in the past,^{26–28} but for the purpose of this work, it is taken to mean a computer “learning” how to best perform some task under a given performance metric via optimization of a set of vector operations. For example, common ML applications are regression problems and classification problems, where the performance metric is usually obvious – e.g. how many items the program correctly sorts in a classification problem, or the mean squared error from a target function in a regression problem. Specifying the correct performance metric and model structure to accomplish a given task is neither deterministic nor necessarily simple,^{29,30} but TensorFlow eases the process with its library of ML algorithms.

Recently, ML methods have been used to improve the accuracy of MD simulations, and some have even achieved configurational accuracy on par with ab-initio methods.^{31–33} However, treating molecular system specific issues like constructing neighbor lists and efficiently using trained models to drive molecular simulation can be difficult and hard reproduce without reviewing computer code.³⁴ This is because implementing a given ML model often requires custom low-level code to achieve learning in the context of MD simulation engines, or because it necessitates an iterative process of generating data, training, and validating, rather than a single ongoing process.^{35,36} HTF can bridge this gap and make the process of connecting ML models with MD simulations easy, transparent, and reproducible via the model specification interface of TensorFlow.

The rest of this paper is broken up into five sections plus

^{a)}Electronic mail: andrew.white@rochester.edu.

some concluding remarks. The first section summarizes the architecture of the HTF package and how it works. The remaining four sections show how ML can be applied to molecular simulation with HTF to a particular task in molecular dynamics, demonstrating online ML in MD, arbitrary collective variable calculations, a use-case of TensorFlow’s automatic differentiation, and learning of coarse-grain forces.

II. METHODS

A more complete description of TensorFlow may be found in Abadi, et al.²⁵ TensorFlow models are built and run in separate steps, similar to how a computer program is compiled and then executed. The models in TensorFlow are expressed as tensor computation graphs where nodes are tensor operations and edges are tensors. The HTF plugin follows the same approach, whereby there is a graph building step and then an execution step. In reality, both of these steps may be performed in a single Python script. We make this conceptual distinction here for ease of discussion.

During the graph building step, placeholder tensors are accessible for neighbor lists, positions, and forces. During the execution step (i.e., running the simulation), they will be populated with their respective values in HOOMD-blue at the current time step. The native HOOMD-blue neighbor list is used. A neighbor list is an $N \times M \times 4$ tensor, where N is the number of particles, M is a pre-set maximum number of neighbors, and the last dimension is x, y, z, w . Here, w_m denotes the m^{th} neighbor’s particle type, and $x_{n,m}, y_{n,m}, z_{n,m}$ are the components of the distance vector to the m^{th} neighbor of the n^{th} particle. The positions, neighbor lists, and forces are provided as possible inputs at each step. However, the format of the neighbor list is changed to a tensor to facilitate further tensor operations in HTF. Additionally, the neighbor list tensor is zero-padded to handle cases where the number of neighbors for a particle is fewer than M . Unfortunately, the optimal choice of M is not well-defined and instead must be considered on a case-by-case basis, based on available computational resources, system size, and ML model complexity. It should be noted that in HOOMD-blue the neighbor list order is not deterministic, resulting in a randomly-ordered neighbor list for each particle at each step.^{1,2} In principle, learning functions that depend only on pairwise particle distance will not be affected by neighbor list order, but for cases where a sorted neighbor list is desired for specific ML objectives, HTF includes a method for enforcing ordered neighbor lists instead. For an example of a deep ML method using neighbor lists in sorted order, see Zhang *et al.*³¹

The tensor computation graph can also output forces and a virial at each step. During the graph building step, these may be computed in the tensor computation graph or automatically calculated from a per-particle or total potential energy tensor. The automatic differentiation computes the forces on particles as:

$$\vec{F}_i = -\frac{\partial U(\mathbf{r})}{\partial \vec{r}_i} - \sum_{i < j} \frac{\partial U(\mathbf{r})}{\partial \vec{r}_{ij}} \quad (1)$$

where $U(\mathbf{r})$ is the potential energy as a function of both positions and/or pairwise distances, $\vec{F}_i$ is the net force on particle i , and $\vec{r}_{ij}$ is the distance vector from particle i to j . HTF computes the per-particle virial contribution from pairwise interactions only:

$$\tau_i = \sum_{i < j} \frac{F_{ij}}{r_{ij}} (\vec{r}_{ij} \otimes \vec{r}_{ij}) \quad (2)$$

Here τ_i is the virial stress tensor of particle i , F_{ij} is the magnitude of the force on particle i from particle j , and $\otimes$ indicates an outer product. If a biasing force in HTF is not pairwise additive, then the virial contribution can be computed by the user to override this default contribution.

During the execution step, while the simulation is running, the TensorFlow graph is executed with the current neighbor lists, positions and forces. Forces can be an input or output. If the TensorFlow graph outputs forces, those will be set in HOOMD-blue to be forces on the particles. Sometimes there is no output, for example if computing a collective variable. Variables in the tensor computation graph allow values to be saved and updated at each step. This allows computed quantities at all stages of the graph to be output. Further, this allows accumulation of values and thus computing quantities like running averages or time-dependent biases. The variables can also be checkpointed and restarted within HTF. During online training with HOOMD-blue, training input can be batched and those batches may be ordered by molecule if desired.

Further details, including software architecture and benchmarking information, can be found in the supporting information (si).

A. Neural Network Force Field (Arbitrary Force Fields)

Neural networks are a powerful class of ML tools whose original theory dates back to the 1950s and 60s.^{37–39} These highly flexible tools have been applied to a variety of diverse tasks in the past, including guiding surveillance technology,^{40,41} tracking visual targets,^{42–44} predicting cancer occurrence in patients,^{45,46} processing medical imaging for enhanced diagnostic accuracy,⁴⁷ controlling the beam of a particle accelerator,⁴⁸ and aiding in finance applications.⁴⁹ They have seen recent use in molecular dynamics simulations, allowing for trained force fields that accurately reproduce experimental conformations³² and dynamical properties in coarse-grained simulations.³¹ Other recent examples include an energy-conserving force field by Chmiela, et al.⁵⁰, later improved to include physical symmetries of molecules,⁵¹ a multi-neural-network-based force field with DFT-level accuracy,⁵² and another neural network force field based on atomic symmetry functions.⁵³

The HTF plugin allows users to designate the structure of neural networks and achieve online training based on HOOMD-blue data during the HOOMD-blue simulation. Any collective variable that can be expressed as a tensor operation on the HOOMD-blue neighbor list or per-particle positions

or forces can be used as training data or neural network input. As a proof of concept, we have trained a neural network to reproduce the Lennard-Jones forces on a 2D simulation of 10000 particles. Training was done online, i.e. during the simulation. The neural network input was the HOOMD-blue neighbor list, and it was trained to output the forces on each particle with an l_2 loss function (mean squared error) and the TensorFlow built-in ADAM optimizer⁵⁴ with a learning rate $\eta = 0.001$, and with TensorFlow’s default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. We employed random dropout at a rate of 20% to avoid overfitting.⁵⁵ The simulation used reduced LJ units with a time step of 0.005, and Langevin integration with $kT = 1.0$. The simulation box was created in HOOMD-blue as a square lattice with lattice constant $a = 2.0$ (100×100 2D box). A complete list of simulation and ML parameters can be found in Table-S2 in si.

The model structure was a dense neural network with three layers: one input layer, one hidden layer, and one output layer. Each layer is size 20 and uses a tanh activation function, except the output layer, which had no activation function. A neural network can be thought of as repeated application of operations $\{\mathcal{F}_i\}$ defined in eq 3, where $\vec{w}_i$ is a weight vector and $\vec{b}_i$ is a bias vector, both of which are unique to each layer, and $\vec{x}_i$ is the vector of input values to layer i . σ_i is some activation function. The limiting case of an activation function is a linear activation, which is the same as “no activation”, where $\sigma(x) = x$. This is rarely used, however, since only neural networks with non-linear activation functions can serve as universal function approximators.⁵⁶ Common activation functions include the sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$, tanh(x), and relu (“rectified linear unit”), $\sigma(x) = \max(0, x)$.

$$\mathcal{F}_i(\vec{x}_i) = \sigma_i(\vec{w}_i \cdot \vec{x}_i + \vec{b}_i) \quad (3)$$

In this case, the neural network’s output is then $\mathcal{F}_2(\mathcal{F}_1(\mathcal{F}_0(\vec{x})))$, with $\sigma_0(x) = \sigma_1(x) = \tanh(x)$ and $\sigma_2(x) = x$. The final layer uses a linear activation function to allow for values of greater magnitude than 1.0. The trained quantities are the weight and bias vectors. The input is $\vec{x} = \vec{r}^{-1}$, the vector of inverse distances between each particle and the particles in its neighbor list.

After a 4000 step equilibration period, the model was trained for 500000 steps, which can be done in minutes using a GPU-enabled compute node (NVIDIA Tesla V100). The l_2 loss of the model over time is shown in Figure 1. The model converged after 450000 time steps with a final loss value of 0.25.

B. Force Matching (Learning)

Bottom-up coarse graining (CG)^{57–59} has been used to model various systems like alkanes⁵⁹, polymers⁶⁰, and biomolecules.⁵⁷ In the bottom-up approach, the CG potentials are derived from the underlying all-atom (AA) simulation. Force matching (FM)^{61–63} is a bottom-up CG approach which aims to match the forces on CG particles as closely as possible to the cumulative forces on their constituent atoms

Figure 1. Using a simple neural network in a 10000 particle Lennard-Jones simulation to learn the Lennard-Jones force field. **A)** The l_2 loss over time of the neural network force field during training (log scale). The l_2 loss is computed as the mean squared difference between the Lennard-Jones forces calculated by HOOMD-blue and those generated by the neural network. The dashed red line shows when cost falls below 1.0, an arbitrary signifier of “good agreement.” **B)** Comparison of the learned potential and the true Lennard-Jones potential, showing recovery of the key features of the Lennard-Jones force field.

in the reference AA simulation.⁵⁷ Besides obtaining the CG potential, another vital step in defining the CG system is to determine the mapping operator, which indicates how atoms in the AA system are grouped into CG particles. The general practice to define a CG mapping operator has historically been driven by chemical intuition.⁶⁴ However, recently there have been efforts toward choosing mapping operators more systemically.^{65–67}

In the following section, let $\mathbf{r}$ represent the coordinates of particles in an all-atom simulation, and let $\mathbf{R}$ represent the corresponding coarse-grained particle positions. The reference mapped force is calculated by eq 4, where F_i^{ref} is the force on the i^{th} CG particle, S_i refers to the subset of particles which are mapped into the i^{th} CG particle and $\vec{F}_i$ is the net force on

Table I. Simulation Parameters for AA Methanol.

Parameter	Value
Dimension	3D
Ensemble	NVT
Thermostat	Nosé-Hoover
Density	778 kg/m ³
time step	2 fs
Temperature	300 K
Force Field	OPLS-AA
Cutoff Radius	10
$N_{\text{molecules}}$	256

the i^{th} particle in the AA model.

$$\mathbf{F}_I^{\text{ref}}(\mathbf{r}) = \sum_{i \in S_I} \vec{F}_i(\mathbf{r}) \quad (4)$$

The FM method finds $\mathbf{F}^{\text{CG}}(\mathbf{R})$ that minimizes the objective function given by eq 5,

$$\chi^2 = \left\langle \frac{1}{3N} \sum_{I=1}^N \left| \mathbf{F}_I^{\text{CG}}(\mathbf{R}) - \mathbf{F}_I^{\text{ref}}(\mathbf{r}) \right|^2 \right\rangle \quad (5)$$

where N is the number of CG particles and the angular brackets denote an ensemble average. In the most common implementation, $\mathbf{F}^{\text{CG}}(\mathbf{R})$ is approximated using cubic splines as the basis set.^{62,68} Previous studies have reported the use of various ML techniques for CG models.^{31,69} Here we model the CG potential using a basis set ($\mathcal{B}$) of 48 Gaussian functions and a repulsive term, $U_{\text{rep}}(R) = u(R - r_0)(R - r_0)^{12}$, where $u()$ is the unit step function, R is the pairwise distance between particles, and r_0 is a fitting parameter. The variance and the height of each of the Gaussian functions in the basis set were trained on-the-fly using mapped CG positions from an atomistic simulation of methanol molecules in HOOMD-blue using the parameters given in Table I. Each methanol molecule was mapped to one CG bead at its center of mass (COM). Corresponding mapped pairwise distances were calculated considering 128 nearest neighbors.

The optimization was done at a learning rate of 0.1 using the ADAM optimizer⁵⁴ by minimizing the objective function given by eq 6

$$\chi^2 = \left\langle \frac{1}{3N} \sum_{I=1}^N \left| \mathbf{F}_I^{\mathcal{B}}(\mathbf{R}) - \mathbf{F}_I^{\text{ref}}(\mathbf{r}) \right|^2 \right\rangle - r_0 \quad (6)$$

where $\mathbf{F}_I^{\mathcal{B}}$ refers to the forces calculated using the basis set and the mapped pairwise distances and r_0 is a fitting parameter for the repulsive function U_{rep} . In eq 6, the first term is the mean squared error between the mapped CG forces from the atomistic simulation and the forces calculated using the the basis set, and the last term pushes the short range repulsive term to the right. A cutoff radius of 12 Å was used for the simulation. After equilibrating the CG system, the production CG simulation was run for 10,000 time steps. Figure 2 shows the

Figure 2. The learned CG potential along with the fit centers of the Gaussian basis set functions for a one bead methanol simulation. A reference potential of mean force (PMF) computed as $-kT \ln[g(r)]$ is plotted. Agreement between CG potential and PMF is not expected, but provides a reference for location of potential energy minima. The repulsion term provides stronger repulsion than is possible with the Gaussian basis set alone.

Figure 3. The center of mass radial distribution function from the AA (Mapped) and the CG simulation of methanol. The one bead CG simulation shows good agreement with a mapped AA simulation.

learned CG potential along with the locations of the Gaussian basis set functions of $\mathcal{B}$. The locations of the minima in the learned CG potential conform well with those reported in previous studies^{70–72} which used conventional tools for FM. Figure 3 compares the COM radial distribution function (RDF) obtained from the AA simulation and that obtained from running a CG simulation using the learned potentials. There is good agreement and this demonstrates the complete process of training and then simulation with trained model.

C. EDS in HTF (Biasing/Automatic differentiation)

Simulations often do not agree with experiments or quantum mechanics results due to force-field or time-scale limitations.⁷³ However, simulations can be biased to agree with experimental observables or reference values.⁷⁴ Experiment Directed Simulation (EDS) is a maximum-entropy biasing technique used to match simulations with some experimental observables.⁷⁴ EDS minimally modifies the free energy surface of a simulation such that average values of simulated collective variables (CVs) match specified average reference values.⁷⁴ If instead of a single average value, an entire free energy distribution along a CV needs to match a target free energy distribution, Experiment Directed Metadynamics can be applied.⁷⁵ EDS can match multiple experimental values in a single iteration of the simulation.⁷⁴ EDS has been previously applied in various systems such as Li ion battery and bead-spring polymer models,⁷⁴ a water model,⁷⁶ a peptide model,⁷³ and in coarse graining techniques.⁷⁷ Dynamical studies of water with EDS showed improved prediction of both biased and unbiased dynamic properties, which are known to be hard to reproduce even with quantum mechanics.⁷⁶ This is possible because EDS minimally modifies the free energy surface and allows accurate prediction of dynamical and structural properties. A recent review on EDS and other maximum entropy methods can be found in Amirkulova and White.⁷⁸ In this work EDS was implemented in the TensorFlow framework in HTF. However, rather than using stochastic gradient descent as in the original implementation,⁷⁴ the TensorFlow Adam optimizer was used with gradients computed following the original EDS paper. In EDS, the potential energy, $U(\vec{r})$ is modified minimally with addition of multiple coupling constants, α_j , as shown in eq 7, where the index j is over the number of CVs to be biased. We desire that the ensemble average of a simulated CV, $f(\vec{r})$, satisfy $\langle f(\vec{r}) \rangle = \hat{f}$, where $\langle f(\vec{r}) \rangle$ is the ensemble average of the CV and $\hat{f}$ is the reference value of that CV. The force of the i^{th} particle, is calculated from the potential given by eq 7 at each time step.

$$U'(\vec{r}, \alpha) = U(\vec{r}) + \sum_j \alpha_j \frac{f_j(\vec{r})}{\hat{f}_j} \quad (7)$$

$$\hat{R}_{COM} = \frac{1}{N} \sum_i^N |\vec{r}_i - \vec{r}_{COM}| \quad (8)$$

A Lennard-Jones particle simulation was performed, and biased with EDS. Mean radius ($\hat{R}_{COM}$), which is defined in eq 8, is an average of absolute differences between each particle position and the center of mass (COM) of all N particles. By using the mean radius as a CV for EDS, we can enforce the simulation on average to form a circle around the COM with a radius of a reference value. We ran EDS and control simulations of Lennard-Jones particles in the canonical ensemble with parameters shown in table II. We assigned random initial velocities to the particles. The only difference between the EDS and control simulations was biasing the potential energy

Table II. EDS Simulation Parameters.

Parameter	Value
Dimension	2D
Ensemble	NVT
Thermostat	Langevin
Box Size	16×16
time step	0.005
kT	1.0
Velocity Randomization Seed	42
Force Field	Lennard-Jones
Cutoff Radius	3.0
$N_{\text{particles}}$	64
Langevin γ	1.0
Lennard-Jones Parameters	
ϵ	1.0 (reduced units)
σ	1.0 (reduced units)
HTF Parameters	
Period	10
M	63
EDS Parameters	
Period	20
Range	0.3
Save Period	1000
$\hat{f}$, reference	4.0

of the EDS simulation at every HTF iteration, which corresponds to “Period” in the EDS Parameters section of table II. The “Range” parameter indicates the range of coupling constant values for the EDS bias, in reduced energy units.

We biased the EDS simulation such that the ensemble average of $\hat{R}_{COM}$ matches a reference value of 4.0, as shown in fig 4. The distribution of particle positions in the simulation box is shown as a heatmap averaged over all all time steps in fig 4. EDS successfully forces the particles to form a circle around the center of the box with a desired radius. The mean distance of particles from the center of mass is shown by a black circle in fig 4-A) with the desired radius of 4.0. The particles in the control simulation are distributed more uniformly as shown in fig 4 (B), where all positions are roughly equally likely.

We have shown that EDS can modify a simulation such that there is an agreement between simulations and chosen observables. This technique can be applied to larger and more complex systems in HOOMD-blue where there may exist a major discrepancy between simulations and experiments.

D. Calculating Scattering Profiles (Arbitrary Collective Variables)

Scattering techniques are used extensively for material characterization and structure determination.⁷⁹ Neutron scattering in particular has applications in the field of condensed matter physics,^{80,81} nuclear physics and nuclear materials research,⁸² biology,⁸³ crystallography, and catalysis.⁸¹ For

Figure 4. Lennard Jones particles are biased to reside inside the circle during EDS. EDS (A) allows particles to be inside of a circle with a radius of 4.0. Control (B) particles do not form a circle. The average distance of particles from their center of mass was 6.1, which is shown as a hypothetical circle in control (B) panel. The circles with radii corresponding to the average radii of particles away from the COM are shown in black. The average radius of 4 during EDS (A) agrees with the reference value of 4, as opposed to the value of 6.1 (B).

this work, we demonstrate that HTF can be used to approximate the neutron scattering profile of a simulation box at every time step, on-the-fly.

The neutron scattering profile for particles can be analytically estimated using Debye’s formula given by eq (9).⁸⁴

$$I(q) = \sum_i \sum_j b_i b_j \frac{\sin(q r_{ij})}{q r_{ij}} \quad (9)$$

Here, b_i and b_j are the coherent scattering lengths for the particles i and j respectively, r_{ij} is the inter-particle distance between particles i and j , and q is the magnitude of $\vec{q}$, the scattering vector or the momentum transfer vector, which is

Figure 5. The neutron scattering profile for 5000 H_2O molecules calculated using HTF. Scattering intensities are calculated for every value of q at each time step on-the-fly and the averaged intensities over all time steps are presented here.

a function of the scattering angle θ and wavelength λ . The data for scattering lengths of different particles is available in literature.⁸⁵

A box of 5000 water molecules was simulated in order to calculate their neutron scattering profile. Figure 5 shows the resultant scattering profile, which was calculated as an average of the scattering intensities at each time step during the simulation. The simulation details are given in Table III. The obtained scattering profile (Figure 5) is in close agreement with the experimentally observed neutron scattering profile for H_2O from Amann-Winkel, et al.⁸⁶

Due to the automatic differentiation included in TensorFlow, a more complex system could be conveniently biased toward an experimentally obtained neutron scattering profile on-the-fly, without necessitating tracking the analytic derivative of eq 9 with respect to r_{ij} of every pair of particles for every value of q .

III. CONCLUSION

We have presented a general tool for utilizing TensorFlow in the molecular simulation engine HOOMD-blue. TensorFlow’s tensor computation graphs are expressive enough to allow force-fields, biasing methods, learning, and collective variable computation within a single framework. HTF enables GPU-accelerated and low-latency use of TensorFlow in HOOMD-blue and thus makes online learning in a simulation possible with little additional effort. The ability to tightly integrate trained ML models in HTF can enable their use in simulations by removing the need for custom implementations and improve reproducibility in the field. The tensor computation graphs allow for transparent and simple model designation with a high degree of customizability, replicability, and efficiency.

Table III. H₂O Simulation Parameters to Calculate Scattering Profile.

Simulation Parameters (LJ reduced units)	
Dimension	3D
Ensemble	NVT
Thermostat	Nosé-Hoover
Box Size	53.1
time step	0.04
Total time steps	0.04
T	298 K
Force Field	OPLS-AA
Water Model	TIP3P
Cutoff Radius	4.0
$N_{\text{molecules}}$	5000
Nosé-Hoover τ	2.0
Neutron Scattering Lengths	
b_H	-3.742
b_O	5.805
HTF Parameters	
Period	10
M	256

The HTF source code is available as specified in the Supporting Information where each of the systems presented in this article are available as examples.

ACKNOWLEDGMENTS

The authors would like to thank Joshua Anderson and Jens Glaser for instructive conversations about the HOOMD-blue architecture. The authors thank the Center for Integrated Research Computing (CIRC) at the University of Rochester for providing computational resources and technical support. This work was supported by the National Science Foundation (CBET-1751471 and CHE-1764415).

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

- ¹J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *J. Comput. Phys.* **227**, 5342–5359 (2008).
- ²J. 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).
- ³H. Lin, S. Lee, L. Sun, M. Spellings, M. Engel, S. C. Glotzer, and C. A. Mirkin, "Clathrate colloidal crystals," *Science* **355**, 931–935 (2017).
- ⁴N. Zhou, A. S. Dudnik, T. I. N. G. Li, E. F. Manley, T. J. Aldrich, P. Guo, H.-C. Liao, Z. Chen, L. X. Chen, R. P. H. Chang, A. Facchetti, M. Olvera

- De La Cruz, and T. J. Marks, "All-Polymer Solar Cell Performance Optimized via Systematic Molecular Weight Tuning of Both Donor and Acceptor Polymers," *J. Am. Chem. Soc.* **138**, 1240–1251 (2015).
- ⁵G. L. Dignon, W. Zheng, R. B. Best, Y. C. Kim, and J. Mittal, "Relation between single-molecule properties and phase behavior of intrinsically disordered proteins," *Proc. Natl. Acad. Sci. U. S. A.* **115**, 9929–9934 (2018).
- ⁶R. E. Guerra, C. P. Kelleher, A. D. Hollingsworth, and P. M. Chaikin, "Freezing on a sphere," *Nature* **554**, 346–350 (2018).
- ⁷E. M. Kujiuk, E. M. Bringa, A. Higginbotham, and C. G. Garino, "GPU Processing of Molecular Dynamics Simulations," in *HPC - LATAM* (Buenos Aires, 2010) pp. 3234–3248.
- ⁸L. Yang, F. Zhang, C.-Z. Wang, K.-M. Ho, and A. Travesset, "Implementation of metal-friendly EAM/FS-type semi-empirical potentials in HOOMD-blue: A GPU-accelerated molecular dynamics software," *J. Comput. Phys.* **359**, 352–360 (2018).
- ⁹T. D. Nguyen, C. L. Phillips, J. A. Anderson, and S. C. Glotzer, "Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units," *Comput. Phys. Commun.* **182**, 2307–2313 (2011).
- ¹⁰A. W. Long, C. L. Phillips, E. Jankowski, and A. L. Ferguson, "Nonlinear machine learning and design of reconfigurable digital colloids," *Soft Matter* **12**, 7119–7135 (2016).
- ¹¹D. N. LeBard, B. G. Levine, R. DeVane, W. Shinoda, and M. L. Klein, "Premicelles and monomer exchange in aqueous surfactant solutions above and below the critical micelle concentration," *Chem. Phys. Lett.* **522**, 38–42 (2012).
- ¹²I. Morozov, A. Kazennov, R. Bystryi, G. Norman, V. Pisarev, and V. Stegailov, "Molecular dynamics simulations of the relaxation processes in the condensed matter on GPUs," *Comput. Phys. Commun.* **182**, 1974–1978 (2011).
- ¹³L. Teich and C. Schroder, "Numerical Investigation of the Magnetodynamics of Self-Organizing Nanoparticle Ensembles: A Hybrid Molecular and Spin Dynamics Approach," *IEEE Trans. Magn.* **51**, 1–4 (2015).
- ¹⁴C. L. Breaux, J. B. Delony, P. J. Ludovice, and C. L. Henderson, "Effect of homopolymer concentration on LER and LWR in block copolymer/homopolymer blends," in *Nov. Patterning Technol. 2018*, Vol. 10584, edited by E. M. Panning and M. I. Sanchez (SPIE, 2018) p. 52.
- ¹⁵F. Benedetti, D. Racko, J. Dorier, Y. Burnier, and A. Stasiak, "Transcription-induced supercoiling explains formation of self-interacting chromatin domains in *S. pombe*," *Nucleic Acids Res.* **45**, 9850–9859 (2017).
- ¹⁶F. Benedetti, A. Japaridze, J. Dorier, D. Racko, R. Kwapich, Y. Burnier, G. Dietler, and A. Stasiak, "Effects of physiological self-crowding of DNA on shape and biological properties of DNA molecules with various levels of supercoiling," *Nucleic Acids Res.* **43**, 2390–2399 (2015).
- ¹⁷C. L. Phillips, J. A. Anderson, and S. C. Glotzer, "Molecular Dynamics Models of Shaped Particles Using Filling Solutions," *Phys. Procedia* **53**, 75–81 (2014).
- ¹⁸B. Trefz and P. Virnau, "Scaling behavior of topologically constrained polymer rings in a melt," *J. Phys. Condens. Matter* **27**, 354110 (2015).
- ¹⁹M. P. Howard, A. Z. Panagiotopoulos, and A. Nikoubashman, "Efficient mesoscale hydrodynamics: Multiparticle collision dynamics with massively parallel GPU acceleration," *Comput. Phys. Commun.* **230**, 10–20 (2018).
- ²⁰B. G. Levine, D. N. LeBard, R. DeVane, W. Shinoda, A. Kohlmeyer, and M. L. Klein, "Micellization Studied by GPU-Accelerated Coarse-Grained Molecular Dynamics," *J. Chem. Theory Comput.* **7**, 4135–4145 (2011).
- ²¹C. L. Phillips, J. A. Anderson, and S. C. Glotzer, "Pseudo-random number generation for Brownian Dynamics and Dissipative Particle Dynamics simulations on GPU devices," *J. Comput. Phys.* **230**, 7191–7201 (2011).
- ²²M. P. Howard, J. A. Anderson, A. Nikoubashman, S. C. Glotzer, and A. Z. Panagiotopoulos, "Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units," *Comput. Phys. Commun.* **203**, 45–52 (2016).
- ²³J. A. Anderson, M. Eric Irrgang, and S. C. Glotzer, "Scalable Metropolis Monte Carlo for simulation of hard shapes," *Comput. Phys. Commun.* **204**, 21–30 (2016).
- ²⁴M. Spellings, R. L. Marson, J. A. Anderson, and S. C. Glotzer, "GPU accelerated Discrete Element Method (DEM) molecular dynamics for conservative, faceted particle simulations," *J. Comput. Phys.* **334**, 460–467 (2017).

- ²⁵M. Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," (2015).
- ²⁶H. Wang, C. Ma, and L. Zhou, "A Brief Review of Machine Learning and Its Application," in *2009 Int. Conf. Inf. Eng. Comput. Sci.* (IEEE, 2009) pp. 1–4.
- ²⁷K. P. Murphy, *Machine learning : a probabilistic perspective* (MIT Press, 2012) p. 1067.
- ²⁸E. Alpaydin, *Introduction to machine learning* (MIT press, 2009).
- ²⁹A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.* **97**, 245–271 (1997).
- ³⁰C. McDonald and L. A. Smith, *Computer science '98 : proceedings of the 21st Australasian Computer Science Conference, ACSC'98, Perth, 4-6 February 1998*, Vol. Volume 20 (Springer, 1998) p. 586.
- ³¹L. 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).
- ³²K. Yao, J. E. Herr, D. W. Toth, R. Mckintyre, and J. Parkhill, "The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics," *Chem. Sci.* **9**, 2261–2269 (2018).
- ³³H. Wang, L. Zhang, J. Han, and W. E, "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics," *Comput. Phys. Commun.* **228**, 178–184 (2018).
- ³⁴K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, "Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies," *J. Chem. Theory Comput.* **9**, 3404–3419 (2013).
- ³⁵M. Rupp, A. Tkatchenko, K.-R. Müller, and O. Anatole Von Lilienfeld, "Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning," *Phys. Rev. Lett.* **108**, 058301 (2012).
- ³⁶V. Botu and R. Ramprasad, "Adaptive Machine Learning Framework to Accelerate Ab Initio Molecular Dynamics," *Int. J. Quantum Chem.*, 1074–1083 (2015).
- ³⁷B. Widrow, *Adaptive "adaline" neuron using chemical "memistors"* (Stanford University, Stanford, 1960) pp. 1553–2.
- ³⁸J. A. Anderson and E. Rosenfeld, *Talking Nets: An Oral History of Neural Networks*, Bradford Books (MIT Press, 2000).
- ³⁹J. Von Neumann and Others, *Cereb. Mech. Behav.*, Vol. 5 (New York: John Wiley & Sons, 1951) pp. 288–326.
- ⁴⁰T. Jan, "Neural network based threat assessment for automated visual surveillance," in *2004 IEEE Int. Jt. Conf. Neural Networks (IEEE Cat. No. 04CH37541)*, Vol. 2 (IEEE, 2004) pp. 1309–1312.
- ⁴¹G. Wang, L. Guo, and H. Duan, "Wavelet neural network using multiple wavelet functions in target threat assessment," *ScientificWorldJournal*, **2013**, 632437 (2013).
- ⁴²Chin-Der Wann and S. Thomopoulos, "Unsupervised learning neural networks with applications to data fusion," in *Proc. 1994 Am. Control Conf. - ACC '94*, Vol. 2 (IEEE, 1994) pp. 1361–1365.
- ⁴³S. Shams, "Neural network optimization for multi-target multi-sensor passive tracking," *Proc. IEEE* **84**, 1442–1457 (1996).
- ⁴⁴S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," in *32nd Int. Conf. Mach. Learn.* (2015) pp. 1–10.
- ⁴⁵M. Karabatak and M. C. Ince, "An expert system for detection of breast cancer based on association rules and neural network," *Expert Syst. Appl.* **36**, 3465–3469 (2009).
- ⁴⁶Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen, "Lung cancer cell identification based on artificial neural network ensembles," *Artif. Intell. Med.* **24**, 25–36 (2002).
- ⁴⁷A. S. Miller, B. H. Blott, and T. K. Hames, "Review of neural network applications in medical imaging and signal processing," *Med. Biol. Eng. Comput.* **30**, 449–464 (1992).
- ⁴⁸B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: applications in industry, business and science," *Commun. ACM* **37**, 93–106 (1994).
- ⁴⁹B. K. Wong, T. A. Bodnovich, and Y. Selvi, "Neural network applications in business: A review and analysis of the literature (1988–1995)," *Decis. Support Syst.* **19**, 301–320 (1997).
- ⁵⁰S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, "Machine learning of accurate energy-conserving molecular force fields," *Sci. Adv.* **3**, e1603015 (2017).
- ⁵¹S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, "Towards exact molecular dynamics simulations with machine-learned force fields," *Nat. Commun.* **9**, 3887 (2018).
- ⁵²J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost," *Chem. Sci.* **8**, 3192–3203 (2017).
- ⁵³Y. Huang, J. Kang, W. A. Goddard, and L.-W. Wang, "Density functional theory based neural network force fields from energy decompositions," *Phys. Rev. B* **99**, 064103 (2019).
- ⁵⁴D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." *CoRR abs/1412.6980* (2014).
- ⁵⁵N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Tech. Rep.* (2014).
- ⁵⁶M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks* **6**, 861–867 (1993).
- ⁵⁷H. I. Ingólfsson, C. A. Lopez, J. J. Uusitalo, D. H. de Jong, S. M. Gopal, X. Periole, and S. J. Marrink, "The Power of Coarse Graining in Biomolecular Simulations," *Wiley Interdisciplinary Reviews: Computational Molecular Science* **4**, 225–248 (2014).
- ⁵⁸P. G. Lafond and S. Izvekov, "Multiscale Coarse-Graining with Effective Polarizabilities: A Fully Bottom-Up Approach," *Journal of Chemical Theory and Computation* **14**, 1873–1886 (2018).
- ⁵⁹N. J. Dunn and W. G. Noid, "Bottom-up coarse-grained models that accurately describe the structure, pressure, and compressibility of molecular liquids," *Journal of Chemical Physics* **143** (2015), 10.1063/1.4937383.
- ⁶⁰V. Sethuraman, S. Mogurampelly, and V. Ganesan, "Multiscale Simulations of Lamellar PS-PEO Block Copolymers Doped with LiPF₆ Ions," *Macromolecules* **50**, 4542–4554 (2017).
- ⁶¹F. Ercolessi and J. B. Adams, "Interatomic Potentials from First-Principles Calculations: The Force-Matching Method," *EPL (Europhysics Letters)* **26**, 583 (1994).
- ⁶²S. Izvekov and G. A. Voth, "Multiscale coarse graining of liquid-state systems," *Journal of Chemical Physics* **123** (2005), 10.1063/1.2038787.
- ⁶³A. Das, L. Lu, H. C. Andersen, and G. A. Voth, "The multiscale coarse-graining method. X. Improved algorithms for constructing coarse-grained potentials for molecular systems," *Journal of Chemical Physics* **136** (2012), 10.1063/1.4705420.
- ⁶⁴J. F. Rudzinski and W. G. Noid, "Investigation of coarse-grained mappings via an iterative generalized Yvon-Born-Green method," *Journal of Physical Chemistry B* **118**, 8295–8312 (2014).
- ⁶⁵M. Chakraborty, C. Xu, and A. D. White, "Encoding and Selecting Coarse-Grain Mapping Operators with Hierarchical Graphs," *J. Chem. Phys.* **149**, 134106 (2018), arXiv:1804.04997.
- ⁶⁶E. Brini, E. a. Algaer, P. Ganguly, C. Li, F. Rodríguez-Ropero, and N. F. a. V. D. Vegt, "Systematic Coarse-Graining Methods for Soft Matter Simulations – A Review," *Soft Matter* **9**, 2108–2119 (2013).
- ⁶⁷Y. L. Chen and M. Habeck, "Data-driven coarse graining of large biomolecular structures," *PLoS ONE* **12**, 1–17 (2017).
- ⁶⁸C. Scherer and D. Andrienko, "Comparison of systematic coarse-graining strategies for soluble conjugated polymers," *European Physical Journal: Special Topics* **225**, 1441–1461 (2016).
- ⁶⁹K. K. Bejagam, S. Singh, Y. An, and S. A. Deshmukh, "Machine-Learned Coarse-Grained Models," *The Journal of Physical Chemistry Letters* **9**, 4667–4672 (2018).
- ⁷⁰V. Rühle, C. Junghans, A. Lukyanov, K. Kremer, and D. Andrienko, "Versatile Object-Oriented Toolkit for Coarse-Graining Applications," *Journal of Chemical Theory and Computation* **5**, 3211–3223 (2009).
- ⁷¹C. Scherer and D. Andrienko, "Understanding three-body contributions to coarse-grained force fields," *Physical Chemistry Chemical Physics* **20**, 22387–22394 (2018).

- ⁷²L. Lu, J. F. Dama, and G. A. Voth, "Fitting coarse-grained distribution functions through an iterative force-matching method," *Journal of Chemical Physics* **139** (2013), 10.1063/1.4811667.
- ⁷³D. B. Amirkulova and A. D. White, "Combining Enhanced Sampling with Experiment Directed Simulation of the GYG peptide," *J. Theor. Comput. Chem.* **17**, 1840007 (2018), arXiv:1804.05100.
- ⁷⁴A. D. White and G. A. Voth, "Efficient and Minimal Method to Bias Molecular Simulations with Experimental Data," *J. Chem. Theory Comput.* (2014).
- ⁷⁵A. D. White, J. F. Dama, and G. A. Voth, "Designing Free Energy Surfaces That Match Experimental Data with Metadynamics," *Journal of Chemical Theory and Computation* **11**, 2451–2460 (2015).
- ⁷⁶A. D. White, C. Knight, G. M. Hocky, and G. A. Voth, "Communication: Improved ab initio molecular dynamics by minimally biasing with experimental data," *J. Chem. Phys.* **146**, 041102 (2017).
- ⁷⁷T. Dannenhoffer-Lafage, A. D. White, and G. A. Voth, "A Direct Method for Incorporating Experimental Data into Multiscale Coarse-Grained Models," *J. Chem. Theory Comput.* **12**, 2144–2153 (2016).
- ⁷⁸D. B. Amirkulova and A. D. White, "Recent Advances in Maximum Entropy Biasing Techniques for Molecular Dynamics," Submitted (2019), <https://arxiv.org/abs/1902.02252>, arXiv:1902.02252.
- ⁷⁹B. Chu and T. Liu, "Characterization of nanoparticles by scattering techniques," *Journal of Nanoparticle Research* **2**, 29–41 (2000).
- ⁸⁰S. F. Parker and P. Collier, "Applications of Neutron Scattering in Catalysis Where atoms are and how they move Neutron Properties and their Applications," *Johnson Matthey Technol. Rev* **60**, 132–144 (2016).
- ⁸¹F. Fernandez-Alonso and D. L. D. L. Price, eds., *Neutron scattering : applications in biology, chemistry, and materials science* (Academic Press, 2017).
- ⁸²S. Vogel, "A review of neutron scattering applications to nuclear materials," *ISRN Mat. Sci.* **2013** (2013).
- ⁸³J. C. Smith, P. Tan, L. Petridis, and L. Hong, "Dynamic Neutron Scattering by Biological Systems," *Annual Review of Biophysics* **47**, 335 – 354 (2018).
- ⁸⁴C. L. Farrow and S. J. L. Billinge, "Relationship between the atomic pair distribution function and small-angle scattering: implications for modeling of nanoparticles," *Acta Crystallographica Section A* **65**, 232–239 (2009).
- ⁸⁵V. F. Sears, "Special Feature Neutron scattering lengths and cross section," *Neutron News* **3**, 26–37 (1992).
- ⁸⁶K. Amann-Winkel, M.-C. Bellissent-Funel, L. E. Bove, T. Loerting, A. Nilsson, A. Paciaroni, D. Schlesinger, and L. Skinner, "X-ray and neutron scattering of water," *Chemical Reviews* **116**, 7570–7589 (2016), pMID: 27195477, <https://doi.org/10.1021/acs.chemrev.5b00663>.
- ⁸⁷M. Bonomi and M. Parrinello, "Enhanced sampling in the well-tempered ensemble," *Physical Review Letters* **104**, 1–4 (2010), arXiv:0910.4914.
- ⁸⁸M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, "Deep learning with dynamic computation graphs," arXiv preprint arXiv:1702.02181 (2017).
- ⁸⁹J. F. Dama, A. V. Sinitskiy, M. McCullagh, J. Weare, B. Roux, A. R. Dinner, and G. A. Voth, "The theory of ultra-coarse-graining. 1. general principles," *Journal of Chemical Theory and Computation* **9**, 2466–2480 (2013), pMID: 26583735.
- ⁹⁰J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "TensorFlow Distributions," ArXiv (2017), arXiv:1711.10604.
- ⁹¹W. Jakob, J. Rhineland, and D. Moldovan, "pybind11 – seamless operability between c++11 and python," (2017), <https://github.com/pybind/pybind11>.
- ⁹²F. Chollet *et al.*, "Keras," <https://keras.io> (2015).
- ⁹³V. Rühle and C. Junghans, "Hybrid Approaches to Coarse-Graining using the VOTCA Package: Liquid Hexane," *Macromolecular Theory and Simulations* **20**, 472–477 (2011).

Supporting Information for A GPU-Accelerated Machine Learning Framework for Molecular Simulation: Hoomd-blue with TensorFlow

R. Barrett,¹ M. Chakraborty,¹ D. Amirkulova,¹ H. Gandhi,¹ and A. White^{1, a)}
Chemical Engineering, University of Rochester.

Source code for our implementation of the methods discussed in the main text can be found at the following GitHub repository: <https://github.com/ur-whitelab/hoomd-tf>.

SOFTWARE ARCHITECTURE

Constructing tensor computation graphs may be confusing, especially because the dimensionalities of the positions and neighbor list tensors are unknown until runtime. However, TensorFlow has a rich set of debugging tools, including TensorBoard,¹ which allows visualization and step-by-step execution of the graph. These are accessible with the HTF plugin. TensorBoard also allows printing, histogramming, and plotting of variables in real time.

HTF is implemented as a multithreading Python module with most of the code written in C++ linked via pybind11.² Data is transferred from HOOMD-blue to TensorFlow simply via GPU-GPU `memcpy` calls. The implementation of HTF in TensorFlow is done by creating custom node types called `hoomd2tf` and `tf2hoomd` which enable reading/writing via the `memcpy` calls to HOOMD-blue data structures. The multithreading is necessary to isolate the TensorFlow and HOOMD-blue GPU contexts, which allows them to run simultaneously on one GPU. The main computation done by HTF is the reshaping of neighbor lists from the ragged-array format of HOOMD-blue into an $N \times M \times 4$ tensor usable by TensorFlow. In addition to GPU-GPU communication, HTF supports MPI parallelization and has a CPU reference implementation.

The MPI parallelization is done via HOOMD-blue’s domain decomposition. Thus, there are N independent TensorFlow instances running, one for each HOOMD-blue domain. This is desirable for pairwise force computation. TensorFlow has the capability of communicating between instances via TCP/IP, which has high latency relative to MPI. This could be used, however, for communicating training data done on batches of frames every so many timesteps, or at the end of a simulation for accumulating/reducing computed values. There exist community-supported MPI implementations of TensorFlow but these have not been explored. The source code of HTF is available as specified in the Supporting Information.

BENCHMARKING

To benchmark HTF, we repeated this process for a range of particle numbers from 100 to 90000 and compared the run times of CPU-only and GPU-enabled HTF to assess its scaling. In Figure S1, we show this benchmarking data, illustrating how low the additional computational cost of training these neural networks can be. Exact values for timesteps per second and N for these trials can be found in Table S1 in the Supporting Information.

N	CPU TPS	GPU TPS
64	6113	4843
144	4042	5119
400	1831	4977
2500	346	3974
10000	86	1781
90000	9	203

Table S1. Median timesteps per second (TPS) over five trials of 5000 training steps of a HOOMD-TF neural network with N Lennard-Jones particles on CPU and GPU.

Figure S1. Median seconds per timestep over five trials of 5000 Training Steps of a HTF Neural Network with N Particles on CPU and GPU. This demonstrates the gain in speed from GPU execution. The linear scaling in particle number, N , shows communication of HOOMD-blue and TensorFlow minimally affects performance. Exact timing values may be found in the Supporting Information.

^{a)}Electronic mail: andrew.white@rochester.edu.

EXAMPLE CODE

To demonstrate the ease and readability of using HTF, we include here some sample code for building a neural network model such as the one used to obtain these results. In this excerpt, *NN* is the size of the neighbor list to be used in HOOMD-blue and **N_hidden_nodes** is the number of nodes in each hidden layer. This code produces a TensorFlow tensor computation graph model for training a neural network force-field. To use this model for inference, the **output_forces** argument would be set to **True**, and it would load the trained model by using the **bootstrap_dir** optional argument to **graph_builder**. This code uses the Keras³ python module to initialize the neural network according to eq ??, with the dimension of $\vec{w}$ set by **N_hidden_nodes** and the operation $\mathcal{F}$ repeated three times.

```
1 import tensorflow as tf
2 import keras
3 import hoomd.tensorflow_plugin as htf
4
5 NN = 64
6 N_hidden_nodes = 5
7 graph = htf.graph_builder(NN, output_forces=False)
8 r_inv = graph.nlist_rinv
9 input_tensor = tf.reshape(r_inv, shape=(-1,1), name
    ='r_inv')
10 input_layer = keras.layers.Input(tensor=
    input_tensor)
11 hidden_layer = keras.layers.Dense(N_hidden_nodes)(
    input_layer)
12 output_layer = keras.layers.Dense(1, input_shape=(
    N_hidden_nodes,))(hidden_layer)
13 nn_energies = tf.reshape(output_layer, [-1, NN])
14 calculated_energies = tf.reduce_sum(nn_energies,
    axis=1, name='calculated_energies')
15 calculated_forces = graph.compute_forces(
    calculated_energies)
16 cost = tf.losses.mean_squared_error(
    calculated_forces, graph.forces)
17 optimizer = tf.train.AdamOptimizer(0.001).minimize(
    cost)
18 graph.save(model_directory=' /tmp/keras_model/',
    out_nodes=[ optimizer])
```

NEURAL NETWORK FORCE FIELD (ARBITRARY FORCE FIELDS)

REFERENCES

- ¹M. Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," (2015).

Table S2. NN Force Field Simulation Parameters.

Simulation Parameters (LJ reduced units)	
Dimension	2D
Ensemble	NVT
Thermostat	Langevin
Box Size	100
time step	0.005
kT	1.0
Force Field	Lennard-Jones
Cutoff Radius	3.0
$N_{\text{particles}}$	10000
Langevin γ	1.0
Lennard-Jones Parameters	
ϵ	1.0
σ	1.0
Neural Network Hyperparameters	
N_{hidden}	1
Nodes per Layer	20
Activation Function	tanh
Optimizer	Adam ⁴
Dropout Rate	0.2
Adam Parameters	
Learning Rate	0.005
β_1	0.9
β_2	0.999
ϵ	10^{-8}
HTF Parameters	
Period	10
M	63

²W. Jakob, J. Rhineland, and D. Moldovan, "pybind11 – seamless operability between c++11 and python," (2017), <https://github.com/pybind/pybind11>.

³F. Chollet *et al.*, "Keras," <https://keras.io> (2015).

⁴D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." CoRR **abs/1412.6980** (2014).

⁵J. C. Smith, P. Tan, L. Petridis, and L. Hong, "Dynamic Neutron Scattering by Biological Systems," Annual Review of Biophysics **47**, 335 – 354 (2018).

⁶B. Chu and T. Liu, "Characterization of nanoparticles by scattering techniques," Journal of Nanoparticle Research **2**, 29–41 (2000).

⁷V. F. Sears, "Special Feature Neutron scattering lengths and cross section," Neutron News **3**, 26–37 (1992).

⁸T. Dannenhoffer-Lafage, A. D. White, and G. A. Voth, "A Direct Method for Incorporating Experimental Data into Multiscale Coarse-Grained Models," J. Chem. Theory Comput. **12**, 2144–2153 (2016).

⁹M. Bonomi and M. Parrinello, "Enhanced sampling in the well-tempered ensemble," Physical Review Letters **104**, 1–4 (2010), arXiv:0910.4914.

¹⁰C. L. Farrow and S. J. L. Billinge, "Relationship between the atomic pair distribution function and small-angle scattering: implications for modeling of nanoparticles," Acta Crystallographica Section A **65**, 232–239 (2009).

¹¹F. Fernandez-Alonso and D. L. D. L. Price, eds., *Neutron scattering : applications in biology, chemistry, and materials science* (Academic Press, 2017).

¹²S. F. Parker and P. Collier, "Applications of Neutron Scattering in Catalysis Where atoms are and how they move Neutron Properties and their Applications," Johnson Matthey Technol. Rev **60**, 132–144 (2016).

- ¹³S. Vogel, "A review of neutron scattering applications to nuclear materials," *ISRN Mat. Sci.* **2013** (2013).
- ¹⁴A. D. White and G. A. Voth, "Efficient and Minimal Method to Bias Molecular Simulations with Experimental Data," *J. Chem. Theory Comput.* (2014).
- ¹⁵A. D. White, C. Knight, G. M. Hocky, and G. A. Voth, "Communication: Improved ab initio molecular dynamics by minimally biasing with experimental data," *J. Chem. Phys.* **146**, 041102 (2017).
- ¹⁶A. D. White, J. F. Dama, and G. A. Voth, "Designing Free Energy Surfaces That Match Experimental Data with Metadynamics," *Journal of Chemical Theory and Computation* **11**, 2451–2460 (2015).
- ¹⁷E. Brini, E. a. Algaer, P. Ganguly, C. Li, F. Rodríguez-Ropero, and N. F. a. V. D. Vegt, "Systematic Coarse-Graining Methods for Soft Matter Simulations – A Review," *Soft Matter* **9**, 2108–2119 (2013).
- ¹⁸P. G. Lafond and S. Izvekov, "Multiscale Coarse-Graining with Effective Polarizabilities: A Fully Bottom-Up Approach," *Journal of Chemical Theory and Computation* **14**, 1873–1886 (2018).
- ¹⁹M. Chakraborty, C. Xu, and A. D. White, "Encoding and Selecting Coarse-Grain Mapping Operators with Hierarchical Graphs," *J. Chem. Phys.* **149**, 134106 (2018), arXiv:1804.04997.
- ²⁰V. Sethuraman, S. Mogurampelly, and V. Ganesan, "Multiscale Simulations of Lamellar PS-PEO Block Copolymers Doped with LiPF₆ Ions," *Macromolecules* **50**, 4542–4554 (2017).
- ²¹N. J. Dunn and W. G. Noid, "Bottom-up coarse-grained models that accurately describe the structure, pressure, and compressibility of molecular liquids," *Journal of Chemical Physics* **143** (2015), 10.1063/1.4937383.
- ²²H. I. Ingólfsson, C. A. Lopez, J. J. Uusitalo, D. H. de Jong, S. M. Gopal, X. Periole, and S. J. Marrink, "The Power of Coarse Graining in Biomolecular Simulations," *Wiley Interdisciplinary Reviews: Computational Molecular Science* **4**, 225–248 (2014).
- ²³Y. L. Chen and M. Habeck, "Data-driven coarse graining of large biomolecular structures," *PLoS ONE* **12**, 1–17 (2017).
- ²⁴F. Ercolessi and J. B. Adams, "Interatomic Potentials from First-Principles Calculations: The Force-Matching Method," *EPL (Europhysics Letters)* **26**, 583 (1994).
- ²⁵K. K. Bejagam, S. Singh, Y. An, and S. A. Deshmukh, "Machine-Learned Coarse-Grained Models," *The Journal of Physical Chemistry Letters* **9**, 4667–4672 (2018).
- ²⁶A. Das, L. Lu, H. C. Andersen, and G. A. Voth, "The multiscale coarse-graining method. X. Improved algorithms for constructing coarse-grained potentials for molecular systems," *Journal of Chemical Physics* **136** (2012), 10.1063/1.4705420.
- ²⁷J. F. Rudzinski and W. G. Noid, "Investigation of coarse-grained mappings via an iterative generalized Yvon-Born-Green method," *Journal of Physical Chemistry B* **118**, 8295–8312 (2014).
- ²⁸C. Scherer and D. Andrienko, "Comparison of systematic coarse-graining strategies for soluble conjugated polymers," *European Physical Journal: Special Topics* **225**, 1441–1461 (2016).
- ²⁹S. Izvekov and G. A. Voth, "Multiscale coarse graining of liquid-state systems," *Journal of Chemical Physics* **123** (2005), 10.1063/1.2038787.
- ³⁰D. B. Amirkulova and A. D. White, "Combining Enhanced Sampling with Experiment Directed Simulation of the GYG peptide," *J. Theor. Comput. Chem.* **17**, 1840007 (2018), arXiv:1804.05100.
- ³¹J. Von Neumann and Others, *Cereb. Mech. Behav.*, Vol. 5 (New York: John Wiley & Sons, 1951) pp. 288–326.
- ³²B. Widrow, *Adaptive "adaline" neuron using chemical "memistors"* (Stanford University, Stanford, 1960) pp. 1553–2.
- ³³J. A. Anderson and E. Rosenfeld, *Talking Nets: An Oral History of Neural Networks*, Bradford Books (MIT Press, 2000).
- ³⁴G. Wang, L. Guo, and H. Duan, "Wavelet neural network using multiple wavelet functions in target threat assessment," *ScientificWorldJournal* **2013**, 632437 (2013).
- ³⁵T. Jan, "Neural network based threat assessment for automated visual surveillance," in *2004 IEEE Int. Jt. Conf. Neural Networks (IEEE Cat. No. 04CH37541)*, Vol. 2 (IEEE, 2004) pp. 1309–1312.
- ³⁶S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," in *32nd Int. Conf. Mach. Learn.* (2015) pp. 1–10.
- ³⁷Chin-Der Wann and S. Thomopoulos, "Unsupervised learning neural networks with applications to data fusion," in *Proc. 1994 Am. Control Conf.* - ACC '94, Vol. 2 (IEEE, 1994) pp. 1361–1365.
- ³⁸S. Shams, "Neural network optimization for multi-target multi-sensor passive tracking," *Proc. IEEE* **84**, 1442–1457 (1996).
- ³⁹M. Karabatak and M. C. Ince, "An expert system for detection of breast cancer based on association rules and neural network," *Expert Syst. Appl.* **36**, 3465–3469 (2009).
- ⁴⁰Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen, "Lung cancer cell identification based on artificial neural network ensembles," *Artif. Intell. Med.* **24**, 25–36 (2002).
- ⁴¹A. S. Miller, B. H. Blott, and T. K. Hames, "Review of neural network applications in medical imaging and signal processing," *Med. Biol. Eng. Comput.* **30**, 449–464 (1992).
- ⁴²K. Yao, J. E. Herr, D. W. Toth, R. Mckintyre, and J. Parkhill, "The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics," *Chem. Sci.* **9**, 2261–2269 (2018).
- ⁴³J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *J. Comput. Phys.* **227**, 5342–5359 (2008).
- ⁴⁴J. 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).
- ⁴⁵H. Lin, S. Lee, L. Sun, M. Spellings, M. Engel, S. C. Glotzer, and C. A. Mirkin, "Clathrate colloidal crystals," *Science* **355**, 931–935 (2017).
- ⁴⁶N. Zhou, A. S. Dudnik, T. I. N. G. Li, E. F. Manley, T. J. Aldrich, P. Guo, H.-C. Liao, Z. Chen, L. X. Chen, R. P. H. Chang, A. Facchetti, M. Olvera De La Cruz, and T. J. Marks, "All-Polymer Solar Cell Performance Optimized via Systematic Molecular Weight Tuning of Both Donor and Acceptor Polymers," *J. Am. Chem. Soc.* **138**, 1240–1251 (2015).
- ⁴⁷R. E. Guerra, C. P. Kelleher, A. D. Hollingsworth, and P. M. Chaikin, "Freezing on a sphere," *Nature* **554**, 346–350 (2018).
- ⁴⁸G. L. Dignon, W. Zheng, R. B. Best, Y. C. Kim, and J. Mittal, "Relation between single-molecule properties and phase behavior of intrinsically disordered proteins," *Proc. Natl. Acad. Sci. U. S. A.* **115**, 9929–9934 (2018).
- ⁴⁹E. M. Kujtiuk, E. M. Bringa, A. Higginbotham, and C. G. Garino, "GP-GPU Processing of Molecular Dynamics Simulations," in *HPC - LATAM* (Buenos Aires, 2010) pp. 3234–3248.
- ⁵⁰L. Yang, F. Zhang, C.-Z. Wang, K.-M. Ho, and A. Travesset, "Implementation of metal-friendly EAM/FS-type semi-empirical potentials in HOOMD-blue: A GPU-accelerated molecular dynamics software," *J. Comput. Phys.* **359**, 352–360 (2018).
- ⁵¹T. D. Nguyen, C. L. Phillips, J. A. Anderson, and S. C. Glotzer, "Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units," *Comput. Phys. Commun.* **182**, 2307–2313 (2011).
- ⁵²A. W. Long, C. L. Phillips, E. Jankowski, and A. L. Ferguson, "Nonlinear machine learning and design of reconfigurable digital colloids," *Soft Matter* **12**, 7119–7135 (2016).
- ⁵³D. N. LeBard, B. G. Levine, R. DeVane, W. Shinoda, and M. L. Klein, "Premicelles and monomer exchange in aqueous surfactant solutions above and below the critical micelle concentration," *Chem. Phys. Lett.* **522**, 38–42 (2012).
- ⁵⁴I. Morozov, A. Kazennov, R. Bystryi, G. Norman, V. Pisarev, and V. Stegailov, "Molecular dynamics simulations of the relaxation processes in the condensed matter on GPUs," *Comput. Phys. Commun.* **182**, 1974–1978 (2011).
- ⁵⁵L. Teich and C. Schroder, "Numerical Investigation of the Magnetodynamics of Self-Organizing Nanoparticle Ensembles: A Hybrid Molecular and Spin Dynamics Approach," *IEEE Trans. Magn.* **51**, 1–4 (2015).
- ⁵⁶C. L. Breaux, J. B. Delony, P. J. Ludovice, and C. L. Henderson, "Effect of homopolymer concentration on LER and LWR in block copolymer/homopolymer blends," in *Nov. Patterning Technol. 2018*, Vol. 10584, edited by E. M. Panning and M. I. Sanchez (SPIE, 2018) p. 52.
- ⁵⁷F. Benedetti, D. Racko, J. Dorier, Y. Burnier, and A. Stasiak, "Transcription-induced supercoiling explains formation of self-interacting chromatin domains in *S. pombe*," *Nucleic Acids Res.* **45**, 9850–9859 (2017).
- ⁵⁸F. Benedetti, A. Japaridze, J. Dorier, D. Racko, R. Kwapich, Y. Burnier, G. Dietler, and A. Stasiak, "Effects of physiological self-crowding of DNA on shape and biological properties of DNA molecules with various levels

- of supercoiling,” *Nucleic Acids Res.* **43**, 2390–2399 (2015).
- ⁵⁹C. L. Phillips, J. A. Anderson, and S. C. Glotzer, “Molecular Dynamics Models of Shaped Particles Using Filling Solutions,” *Phys. Procedia* **53**, 75–81 (2014).
 - ⁶⁰B. Trefz and P. Virnau, “Scaling behavior of topologically constrained polymer rings in a melt,” *J. Phys. Condens. Matter* **27**, 354110 (2015).
 - ⁶¹M. P. Howard, A. Z. Panagiotopoulos, and A. Nikoubashman, “Efficient mesoscale hydrodynamics: Multiparticle collision dynamics with massively parallel GPU acceleration,” *Comput. Phys. Commun.* **230**, 10–20 (2018).
 - ⁶²B. G. Levine, D. N. LeBard, R. DeVane, W. Shinoda, A. Kohlmeyer, and M. L. Klein, “Micellization Studied by GPU-Accelerated Coarse-Grained Molecular Dynamics,” *J. Chem. Theory Comput.* **7**, 4135–4145 (2011).
 - ⁶³C. L. Phillips, J. A. Anderson, and S. C. Glotzer, “Pseudo-random number generation for Brownian Dynamics and Dissipative Particle Dynamics simulations on GPU devices,” *J. Comput. Phys.* **230**, 7191–7201 (2011).
 - ⁶⁴M. P. Howard, J. A. Anderson, A. Nikoubashman, S. C. Glotzer, and A. Z. Panagiotopoulos, “Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units,” *Comput. Phys. Commun.* **203**, 45–52 (2016).
 - ⁶⁵J. A. Anderson, M. Eric Irrgang, and S. C. Glotzer, “Scalable Metropolis Monte Carlo for simulation of hard shapes,” *Comput. Phys. Commun.* **204**, 21–30 (2016).
 - ⁶⁶M. Spellings, R. L. Marson, J. A. Anderson, and S. C. Glotzer, “GPU accelerated Discrete Element Method (DEM) molecular dynamics for conservative, faceted particle simulations,” *J. Comput. Phys.* **334**, 460–467 (2017).
 - ⁶⁷E. Alpaydin, *Introduction to machine learning* (MIT press, 2009).
 - ⁶⁸H. Wang, C. Ma, and L. Zhou, “A Brief Review of Machine Learning and Its Application,” in *2009 Int. Conf. Inf. Eng. Comput. Sci.* (IEEE, 2009) pp. 1–4.
 - ⁶⁹K. P. Murphy, *Machine learning : a probabilistic perspective* (MIT Press, 2012) p. 1067.
 - ⁷⁰A. L. Blum and P. Langley, “Selection of relevant features and examples in machine learning,” *Artif. Intell.* **97**, 245–271 (1997).
 - ⁷¹C. McDonald and L. A. Smith, *Computer science '98 : proceedings of the 21st Australasian Computer Science Conference, ACSC'98, Perth, 4-6 February 1998*, Vol. Volume 20 (Springer, 1998) p. 586.
 - ⁷²M. Rupp, A. Tkatchenko, K.-R. Müller, and O. Anatole Von Lilienfeld, “Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning,” *Phys. Rev. Lett.* **108**, 058301 (2012).
 - ⁷³V. Botu and R. Ramprasad, “Adaptive Machine Learning Framework to Accelerate Ab Initio Molecular Dynamics,” *Int. J. Quantum Chem.*, 1074–1083 (2015).
 - ⁷⁴B. Widrow, D. E. Rumelhart, and M. A. Lehr, “Neural networks: applications in industry, business and science,” *Commun. ACM* **37**, 93–106 (1994).
 - ⁷⁵B. K. Wong, T. A. Bodnovich, and Y. Selvi, “Neural network applications in business: A review and analysis of the literature (1988–1995),” *Decis. Support Syst.* **19**, 301–320 (1997).
 - ⁷⁶K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, “Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies,” *J. Chem. Theory Comput.* **9**, 3404–3419 (2013).
 - ⁷⁷H. Wang, L. Zhang, J. Han, and W. E, “DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics,” *Comput. Phys. Commun.* **228**, 178–184 (2018).
 - ⁷⁸D. B. Amirkulova and A. D. White, “Recent Advances in Maximum Entropy Biasing Techniques for Molecular Dynamics,” Submitted (2019), <https://arxiv.org/abs/1902.02252>, arXiv:1902.02252.
 - ⁷⁹M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, “Deep learning with dynamic computation graphs,” arXiv preprint arXiv:1702.02181 (2017).
 - ⁸⁰J. F. Dama, A. V. Sinitskiy, M. McCullagh, J. Weare, B. Roux, A. R. Dinner, and G. A. Voth, “The theory of ultra-coarse-graining. 1. general principles,” *Journal of Chemical Theory and Computation* **9**, 2466–2480 (2013), pMID: 26583735.
 - ⁸¹J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, “TensorFlow Distributions,” ArXiv (2017), arXiv:1711.10604.
 - ⁸²M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural Networks* **6**, 861 – 867 (1993).
 - ⁸³N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Tech. Rep.* (2014).
 - ⁸⁴Y. Huang, J. Kang, W. A. Goddard, and L.-W. Wang, “Density functional theory based neural network force fields from energy decompositions,” *Phys. Rev. B* **99**, 064103 (2019).
 - ⁸⁵S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, “Machine learning of accurate energy-conserving molecular force fields,” *Sci. Adv.* **3**, e1603015 (2017).
 - ⁸⁶S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, “Towards exact molecular dynamics simulations with machine-learned force fields,” *Nat. Commun.* **9**, 3887 (2018).
 - ⁸⁷J. S. Smith, O. Isayev, and A. E. Roitberg, “ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost,” *Chem. Sci.* **8**, 3192–3203 (2017).
 - ⁸⁸C. Scherer and D. Andrienko, “Understanding three-body contributions to coarse-grained force fields,” *Physical Chemistry Chemical Physics* **20**, 22387–22394 (2018).
 - ⁸⁹L. Lu, J. F. Dama, and G. A. Voth, “Fitting coarse-grained distribution functions through an iterative force-matching method,” *Journal of Chemical Physics* **139** (2013), 10.1063/1.4811667.
 - ⁹⁰V. Rühle, C. Junghans, A. Lukyanov, K. Kremer, and D. Andrienko, “Versatile Object-Oriented Toolkit for Coarse-Graining Applications,” *Journal of Chemical Theory and Computation* **5**, 3211–3223 (2009).
 - ⁹¹V. Rühle and C. Junghans, “Hybrid Approaches to Coarse-Graining using the VOTCA Package: Liquid Hexane,” *Macromolecular Theory and Simulations* **20**, 472–477 (2011).
 - ⁹²K. Amann-Winkel, M.-C. Bellissent-Funel, L. E. Bove, T. Loerting, A. Nilsson, A. Paciaroni, D. Schlesinger, and L. Skinner, “X-ray and neutron scattering of water,” *Chemical Reviews* **116**, 7570–7589 (2016), pMID: 27195477, <https://doi.org/10.1021/acs.chemrev.5b00663>.
 - ⁹³L. 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).