Interaction of complex particles: A framework for the rapid and accurate approximation of pair potentials using neural networks

Gusten Isfeldt
Department of Engineering Mechanics, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden and Department of Fibre and Polymer Technology, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden

Fredrik Lundell
Department of Engineering Mechanics, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden and Wallenberg Wood Science Center, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden

Jakob Wohlert*
Department of Fibre and Polymer Technology, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden and Wallenberg Wood Science Center, KTH Royal Institute of Technology, SE10044 Stockholm, Sweden

(Received 24 June 2024; accepted 16 October 2024; published 12 November 2024)

Motivated by the limitations of conventional coarse-grained molecular dynamics for simulation of large systems of nanoparticles and the challenges in efficiently representing general pair potentials for rigid bodies, we present a method for approximating general rigid body pair potentials based on a specialized type of deep neural network that maintains essential properties, such as conservation of energy and invariance to the chosen origins of the particles. The network uses a specialized geometric abstraction layer to convert the relative coordinates of the rigid bodies to input more suitable to a conventional artificial neural network, which is trained together with the specialized layer. This results in geometric representations of the particles optimized for the specific potential. The network can be trained directly on scalar values to fit a model without explicit gradient and then used to efficiently evaluate the force and torque on the particles resulting from the potential. The concept is demonstrated with an atomistic interaction model for carbon nanotubes and the resulting model is compared with a common type of coarse-grained model optimized for the same potential, with even very small networks comparing favourably and larger networks achieving up to two orders of magnitude lower cost. The sensitivity to noise in the training data is investigated and the model is found to strongly reject noise up to 12.5% given a dataset of 10^7 samples. The performance of a proof-of-concept implementation is demonstrated on a variety of hardware, showing the models viability for large-scale simulations. Furthermore, generalization to soft bodies and potentials for polydisperse systems are discussed.

DOI: 10.1103/PhysRevE.110.055305

I. INTRODUCTION

Understanding the behavior of nanoparticles in dispersions is important in many contexts, as the structures they form determine both macroscopic properties of the dispersions themselves and the materials that can be assembled from them in processes like spinning, spraying, and drying [1,2]. Experimental methods for characterizing these structures, such as small-angle x-ray scattering, typically provide limited and indirect information that is difficult to interpret without a strong hypothesis. Even with complete structural data, fully understanding the mechanisms that govern the assembly processes through this top-down approach might not be possible.

Particle simulations provide another perspective that allows more direct observation of structures as they form and control over the mechanisms involved. Along with diffusion and externally applied forces, the interactions between particles are essential to the assembly processes. Due to the high surface-to-volume ratios at the nanoscale, molecular interactions at interfaces between particles cannot be neglected. Furthermore, the nonadditive nature of fundamental interactions at this scale, as described by Batista *et al.* [3], and geometric complexity of some types of nanoparticles, make mean field approaches like the Derjaguin-Landau-Verwey-Overbeek (DLVO) theory [4,5] inapplicable. These problems can sometimes be overcome with atomistic simulation, which can in many cases accurately predict behavior in processes like self-assembly [3]. However, many observable phenomena governed by these interactions occur at scales far beyond what can feasibly be simulated atomistically. A common approach to solving this issue is the use of coarse-grained molecular dynamics, where atomistic models are reduced such that each nanoparticle can be represented by a smaller number of point particles. This allows the use of conventional molecular

*Contact author: jacke@kth.se

dynamics software, but also inherently limits the model in some ways. The particle interaction can only be a sum of point−point interactions, and since each point particle needs to be tracked, there is a lot of redundant state, especially in the case of relatively rigid particles. Internal degrees of freedom in the particle representation can also limit the possible length of time steps in a simulation to avoid numerical instability and nonphysical oscillations, decreasing the accessible timescales.

Modeling the particles as rigid bodies solves these issues, but introduces a new set of problems. The relative coordinate space of point particles is just $\mathbb{R}1$, but the relative coordinate space of general rigid bodies is significantly more complex, having both a relative position and orientation, for a total of six degrees of freedom. It is difficult to interpolate in this coordinate space using conventional methods. While several simplified models for special cases such as ellipsoids [6] exist and have been used for qualitative studies of certain properties [7], quantitatively relating these to real systems becomes difficult. Accurately and efficiently representing the more general rigid body interaction required for many types of nanoparticles remains challenging.

Assuming that an accurate, but expensive to evaluate, interaction model exists, interpolation of the interaction potential becomes an attractive option, as this would enable its application in larger systems where direct evaluation for all interactions would be infeasible. However, interpolation in this coordinate space is difficult due to the combination of both position and orientation. The simplest general representation is an $\mathbb{R}3$ vector $\mathbf{r}$ for the position and a unit quaternion $\mathbf{q}$ for orientation. Many methods for interpolation in $\mathbb{R}3$ exist, and while less straightforward, it is also possible to interpolate functions on the surface of the hypersphere that the unit quaternion represents [8]. The methods of interpolation are, however, difficult to combine, and the number of data points to densely represent a function in the $(\mathbf{r}, \mathbf{q})$ space with adequate resolution is large. For every point in the relevant $\mathbb{R}3$ domain, a function on the hypersphere must be defined. Assuming a grid of $n \times n \times n$ points with n^3 points on the surface of the hypersphere at each grid point, the cost of increased resolution scales with $\mathcal{O}(n^6)$, which makes higher resolutions, require not only much more input data, but also more working memory for efficient evaluation. Furthermore, depending on the geometry of the interacting particles, the required angular resolution to avoid particles passing through each other can locally be quite high. Take, for example, high aspect-ratio rods, with origins defined at their midpoints (Fig. 1). If the point of contact is close to the end of one particle, the minimum angular resolution to ensure at least one data point with an overlapping, and thus high energy, configuration, to prevent the rod from rotating through the other particle will be $\sim L/2D$ points per radian, scaling linearly with the aspect ratio. Given the above scaling, an aspect ratio of $10:1$ would require a minimum of $\sim 10^9$ data points. Since this data is likely to be quite sparse in terms of actual close interaction, in most cases it should be possible to find a representation that is much more compact.

Approximating the interaction with a neural network stands out as a possible solution, and similar machine learning methods have been used to apply density functional theory−based potentials in atomistic simulations. An early example of this is the work of Behler *et al.* [9], who used a method that applied neural networks to values describing the geometric environment of each individual atom. Later, Chmiela *et al.* [10] developed an efficient gradient machine learning approach, which was able to achieve good results for small molecules with relatively small datasets. This was then improved by Dragoni *et al.* [11] with the added use of symmetries in the molecule to further reduce the amount of training data needed. More recently, Kabylda *et al.* [12] used a optimized descriptor scheme to describe molecular configurations of extended molecules for more efficient application of machine learning potentials. For rigid bodies, methods like Lagrangian graph neural networks have also been applied by Bhattoo *et al.* [13] to simulate systems of connected particles, such as chains.

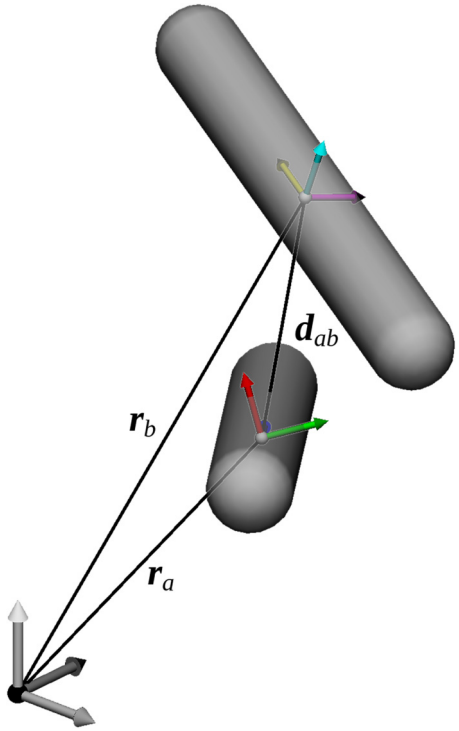


FIG. 1. Two rod-shaped rigid bodies, each with a position and orientation in the global coordinate system.

This study explores the use of neural networks for the application of approximating rigid body pair potentials to enable large-scale simulation of nanoparticles. Through systematic consideration of desirable properties for this specific application, a specialized network architecture is gradually designed that improves performance and guarantees coordinate invariance and physical properties such as energy conservation. In addition, the network model is compared with a conventional coarse-grained model for carbon nanotubes (CNTs).

II. APPLICATION OF THE INTERACTION MODEL

The use of a neural network in this context, as shown in Fig. 2, can broadly be split into two parts: preprocessing, which happens before simulation and results in a trained model, and interaction, which uses the trained model to

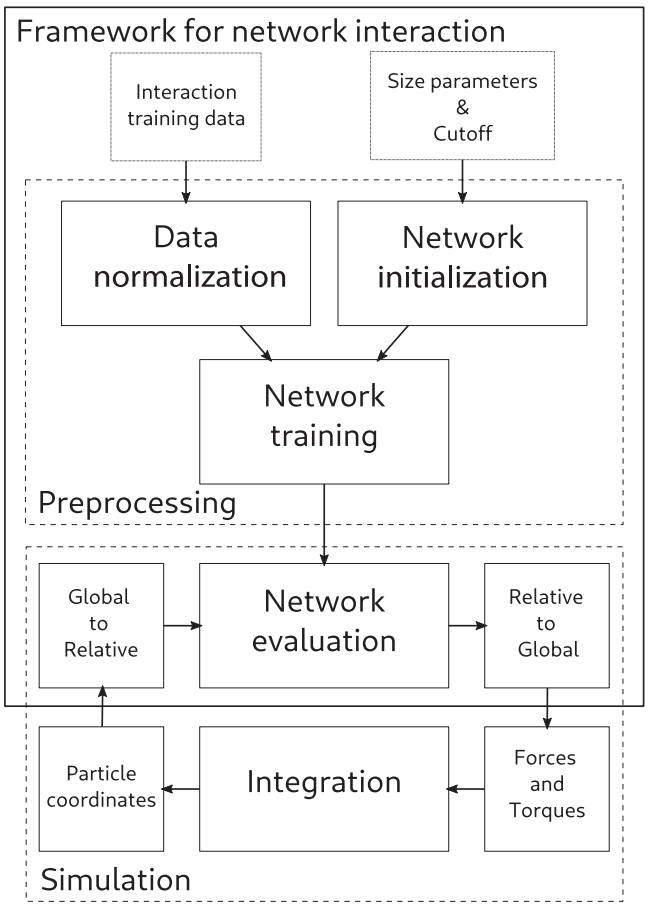


FIG. 2. Schematic showing the different process components in the context of a dynamic simulation, separated into prepossessing, which happens once for each type of interaction, and the simulation, where forces and torques are generated by evaluating the network for each pair of particles and integrated to produce new coordinates at each time step.

compute the interaction between particles, resulting in forces and torques that can be used to time step a simulation.

Since the network operates on relative coordinates, to apply the network to the interaction between two particles in a simulation, their global coordinates $(\boldsymbol{r}_a, \boldsymbol{q}_a)$ and $(\boldsymbol{r}_b, \boldsymbol{q}_b)$ must be converted to relative coordinates, which can be defined as the distance vector and relative orientation

$$\begin{aligned} \boldsymbol{d}_{ab} &= \boldsymbol{q}_a^{-1}(\boldsymbol{r}_b - \boldsymbol{r}_a)\boldsymbol{q}_a \\ \boldsymbol{q}_{ab} &= \boldsymbol{q}_a^{-1}\boldsymbol{q}_b. \end{aligned} \tag{1}$$

The force and torque output from the network $(\boldsymbol{f}_{ab}, \boldsymbol{t}_{ab})$ must then similarly be converted back to force and torque vectors acting on both particles in the global coordinate system. For particle a, the vectors are just rotated back

$$\begin{aligned} \boldsymbol{f}_a &= \boldsymbol{q}_a \boldsymbol{f}_{ab} \boldsymbol{q}_a^{-1} \\ \boldsymbol{t}_a &= \boldsymbol{q}_a \boldsymbol{t}_{ab} \boldsymbol{q}_a^{-1}, \end{aligned} \tag{2}$$

and for particle b, the reaction force and torque

$$\begin{aligned} \boldsymbol{f}_b &= -\boldsymbol{f}_a \\ \boldsymbol{t}_b &= -\boldsymbol{t}_a + (\boldsymbol{r}_b - \boldsymbol{r}_a) \times \boldsymbol{f}_a \end{aligned} \tag{3}$$

are calculated. The forces and torques on each particle from different interactions can then be summed in the global coordinate system.

III. NETWORK ARCHITECTURE

The naive approach to using a neural network for approximating torque and force vectors, shown in Fig. 3(a), simply employs a fully connected deep neural network, as described by Nielsen [14], fitted to each vector component. There are several problems with this naive approach that render it unsuitable for the intended application, but which can be addressed by developing a specialized network structure, that by construction guarantees certain properties of the approximation.

A. Force and torque from back-propagation

While it is possible to directly fit the network to approximate each component of the force and torque vectors, this approach is problematic. Since each of the fitted functions is arbitrary, in general energy will not be conserved in the interaction. The network will also need to be larger, with more parameters, requiring more training data. Additionally, it requires force and torque data for training, instead of just a potential, which might be difficult to compute for some models. It is preferable to instead fit the network to the potential itself, and then differentiate the network to get the force and torque, as seen in Fig. 3(b).

In machine learning, back-propagation is most commonly used to calculate the gradient of the cost function of the network parameters, to optimize the parameters through gradient descent. It can, however, also be used to calculate the gradient of the network output, in this case, the potential itself. The gradient of the relative position and orientation can be converted into force and torque vectors as

$$\boldsymbol{f}_{ab} = \frac{\partial U}{\partial \boldsymbol{d}_{ab}} \tag{4}$$

and

$$\boldsymbol{t}_{ab} = \boldsymbol{f}_{ab} \times \boldsymbol{d}_{ab} - \frac{1}{2}\boldsymbol{Q}\frac{\partial U}{\partial \boldsymbol{q}_{ab}}, \tag{5}$$

respectively, where

$$\boldsymbol{Q} = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix}. \tag{6}$$

.

B. Geometric abstraction layer

In theory, any general purpose function approximator can be fitted to a rigid body potential. However, the nature of the relative coordinate space, with its translational and rotational components, lead to poor behavior for some relatively common types of potentials. An example of this is the interaction of high aspect-ratio rods. For contacts far from the origins of the respective rods, the sensitivity to the relative orientation becomes extremely high, leading to locally large

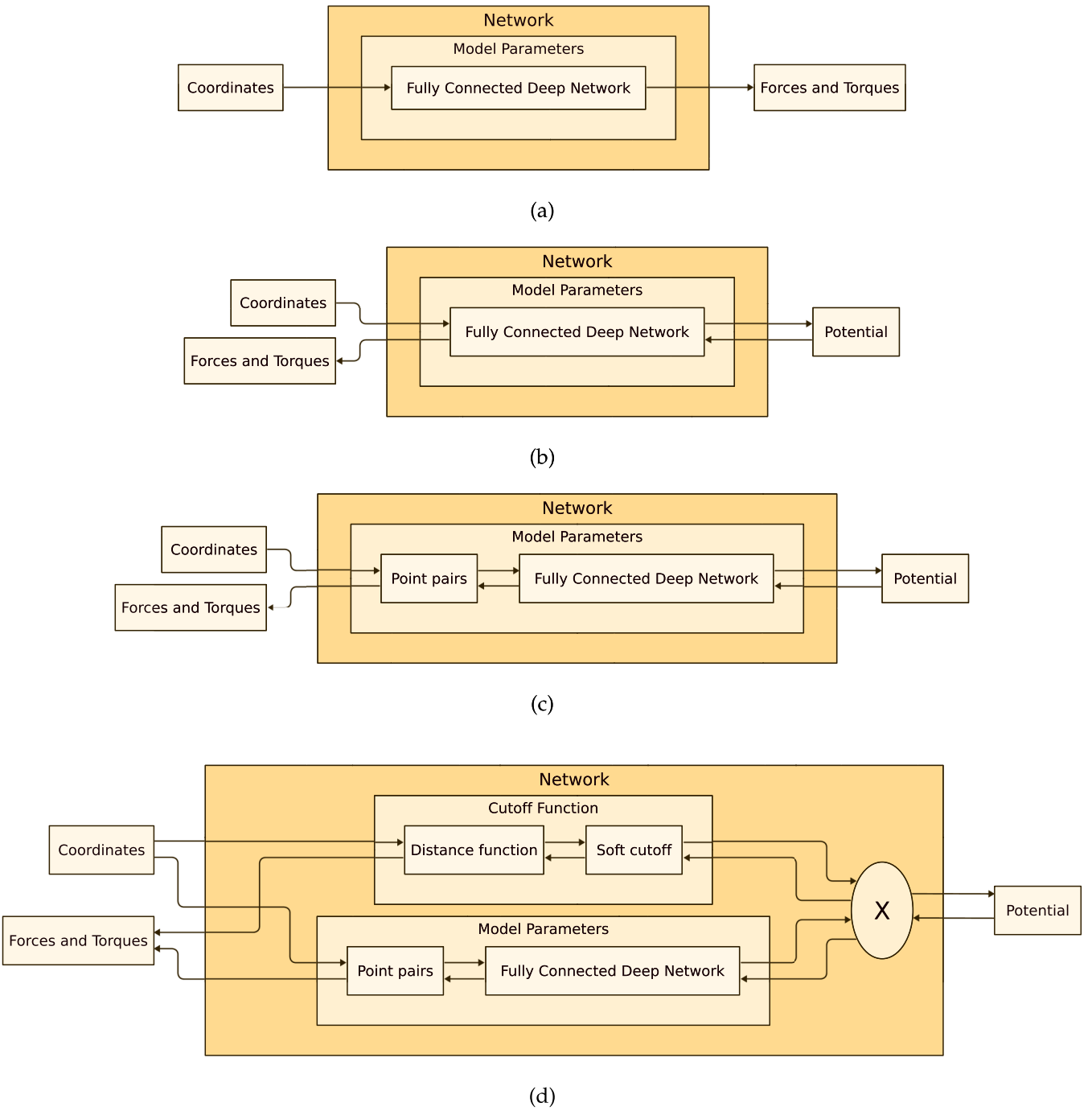


FIG. 3. Network architectures of increasing complexity and specialization, starting from the naive solution and gradually adding features to improve the properties of the model. Arrows show the flow of information through the network during evaluation, starting from the coordinates and ending at the forces and torques. (a) Network trained directly on force and torque vectors. (b) Network trained on potential, force and torque calculated through back-propagation to ensure energy conservation. (c) Network with geometric abstraction layer that transforms coordinates with an optimised point representation that provides coordinate invariance and improves convergence. (d) Network with a smooth, finite cutoff which enables O(n) scaling in systems with many particles.

gradients that are hard to fit accurately. The fitting of the model also becomes dependent on the origins chosen for each particle, since these will affect how easy it is to fit an approximation.

To avoid this problem, the coordinates can be converted to a form that more directly represents the geometry of the interaction. Here, this is achieved by going back to a point representation, where distances are easily defined. Choosing a point in the coordinate system of each particle, $\boldsymbol{p}_{ai}$ and $\boldsymbol{p}_{bi}$, the distance between these points

$$d_i(\boldsymbol{d}_{ab}, \boldsymbol{q}_{ab}) = \left|\boldsymbol{d}_{ab} - \boldsymbol{p}_{ai} + \boldsymbol{q}_{ab}\boldsymbol{p}_{bi}\boldsymbol{q}_{ab}^{-1}\right| \tag{7}$$

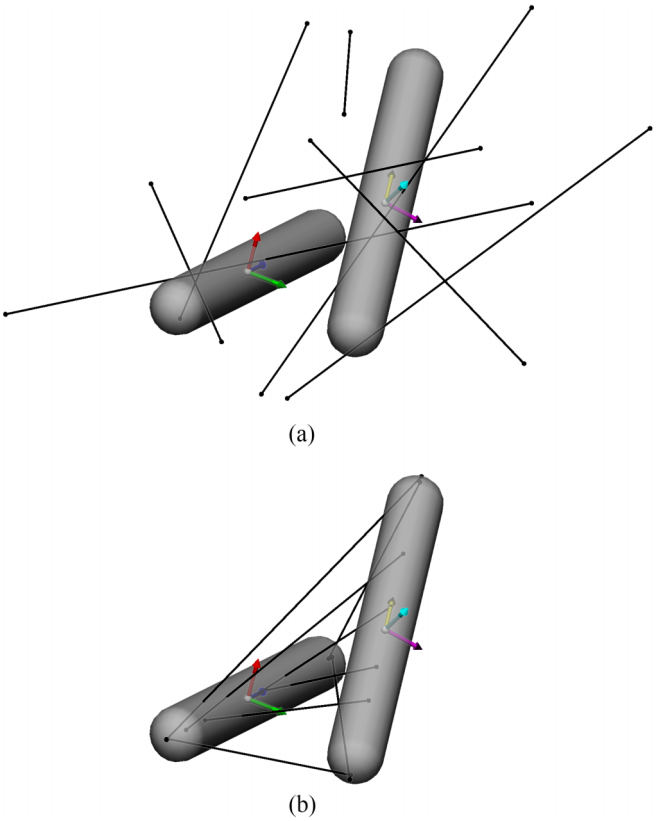


FIG. 4. During the optimization process, the points move to better represent the particle geometry. For these cylindrical particles, the points quickly converge to the axes of the particles, thereby finding the symmetry in the interaction. Contrary to what might be expected, the optimized configuration includes points that lie outside the envelope of the particles. Crucially, the ends of the particles are connected, which appears to be the most important distance measures. (a) Before optimization and (b) After optimization.

becomes a reduced relative coordinate for the rigid bodies. By itself, one such pair of points is not very useful since so much information is lost in the reduction, but combining several such pairs (Fig. 4) results in a flexible coordinate representation. The optimal set of pairs depends on the underlying geometry of the interaction. Viewing this conversion as the first layer in an artificial neural network, as seen in Fig. 3(c), with the point positions as parameters, is convenient since this allows for the simultaneous optimization of both the geometric representation and the potential approximation based on it. Since the point positions are optimized, the ability to fit the model becomes independent on the origins chosen for each particle.

The local gradient of the potential for each point, calculated through back-propagation, can be interpreted as a force acting on it. Adding the forces and resultant torques,

$$\boldsymbol{f}_{ab} = -\sum_i \frac{\partial V}{\partial \boldsymbol{p}_{ai}} \tag{8}$$

$$\boldsymbol{t}_{ab} = -\sum_i \boldsymbol{p}_{ai} \times \frac{\partial V}{\partial \boldsymbol{p}_{ai}} \tag{9}$$

gives the force and torque between the rigid bodies. The distance between two points is problematic to use directly, because the gradient is undefined when the points coincide, which may happen since points can be outside the boundaries of the particles. This can be solved by separating them by a fixed distance w in a fourth dimension, to create a modified distance

$$d_i^* = \sqrt{w^2 + d_i^2}, \tag{10}$$

which is smooth over the whole domain, for all $w \neq 0$. Since w is part of the particle representation, it should be selected based on the scale of the particle geometry. A very small w will have a negligible smoothing effect, and a very large w will smooth the function to the point that it cannot fit to training data. It should, therefore, be chosen to be a small but relevant particle length scale for the potential it is being fitted to.

C. Smooth cutoff

For practical application of a pair potential on a large system of particles, a cutoff is needed to limit the number of interactions that need to be evaluated and avoid $\mathcal{O}(n^2)$ scaling with the number of particles. To conserve energy in a system with a finite cutoff, the potential must go smoothly to zero at the cutoff. This property can be ensured by multiplying the output of the network with a function with these properties, as seen in Fig. 3(d). If this smooth cutoff is included in the model during optimization, the parameters are optimized to partially compensate for the shape of the cutoff function while maintaining the desired boundary condition.

Preferably, the cutoff should be set such that all relevant parts of the interaction space are kept while excluding as much as possible of the space where the interaction is negligible. For low aspect-ratio particles, a spherical cutoff is acceptable, but for higher aspect-ratio particles, a spherical cutoff would result in many evaluations where the particles are far apart. A viable option for rods is to use the minimum distance between line segments instead of the center-center distance. The minimum distance can be found by solving a small quadratic optimization problem, as shown in Appendix B 1.

To achieve a smooth transition to zero the function,

$$\phi(d) = \begin{cases} d \leqslant 1 \to d^{2n}(n(d^2-1)-1)) + 1 \\ d > 1 \to 0 \end{cases} \tag{11}$$

is applied to the calculated normalized distance $d_{\text{norm}} = d/d_{\text{cutoff}}$. For integer $n > 0$, ϕ goes smoothly from one to zero as the cutoff is approached. As seen in Fig. 5, higher n give a more sudden cutoff and keeps the value approximately constant in a larger part of the domain.

D. Cost and data normalization

Given the intended application of the network, accuracy at all inputs are not of equal importance. Since high-energy states are less likely to occur during simulation of a sparse particle system, they should be given lower weights in the optimiszation to minimize total simulation error. This is also helpful since many potentials commonly used in molecular dynamics, such as the Lennard-Jones and Coloumb potentials, are singular, which would be impossible to accurately replicate with a network using nonsingular activation functions.

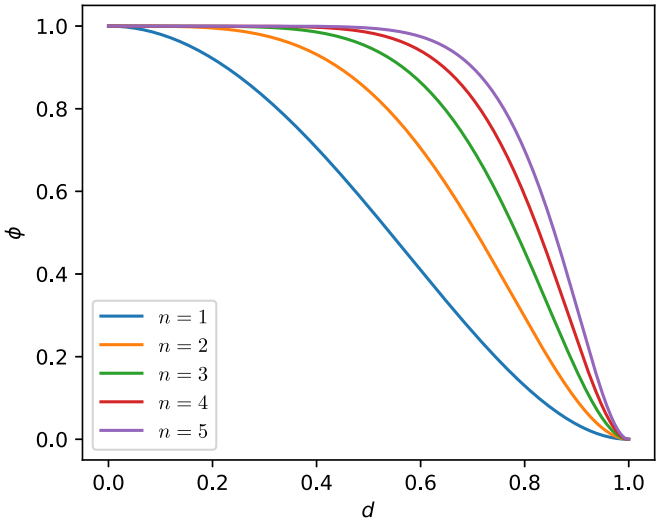


FIG. 5. The scaling factor ϕ, from Eq. (11) as a function of the normalized distance d for a smooth cutoff.

This weighting can be achieved by using a suitable cost function in the optimization. The cost function quantifies how poor the fit is given the exact values V_{ref} and approximated values V_{net} at a point, and its gradient is used in the optimization process. High-energy configurations can be given lower weight by introducing a threshold value V_{max}, over which the cost function becomes less sensitive to deviations. To be useful for gradient descent optimization, the function also needs to have a usable gradient over the whole domain. The function

$$C(V_{\text{ref}}, V_{\text{net}}) = \left(\ln\left(1 + e^{\frac{-V_{\text{ref}}}{V_{\text{max}}}}\right) - \ln\left(1 + e^{\frac{-V_{\text{net}}}{V_{\text{max}}}}\right)\right)^2 \quad (12)$$

was derived by modifying the common quadratic cost function to use a soft truncation of both the reference potential and the approximated potential. As seen in Fig. 6, this results in a function that gradually becomes less sensitive to deviations

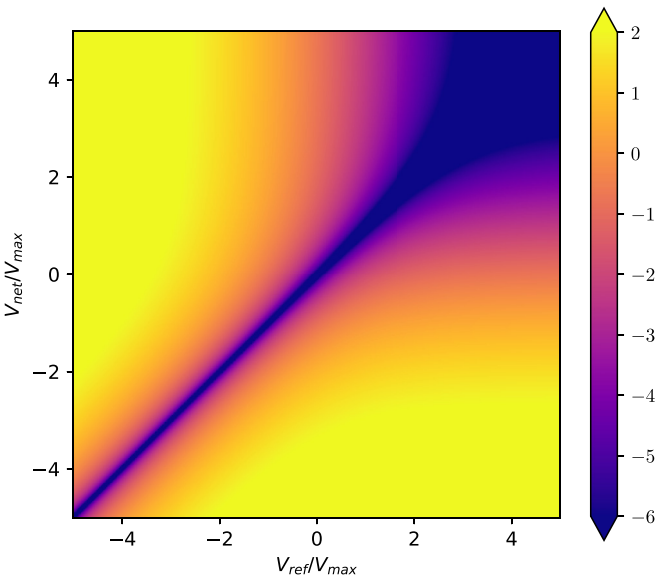


FIG. 6. Logarithm of cost as a function of V_{ref} and V_{net}, normalized to V_{max}.

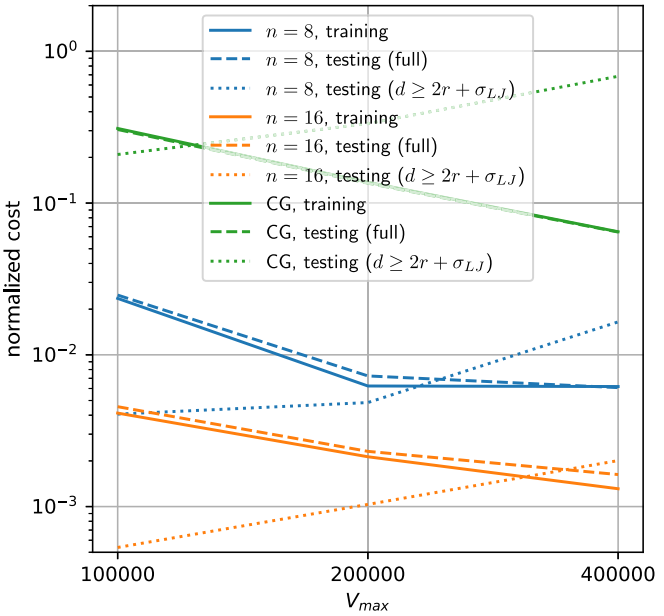


FIG. 7. Convergence for different data sets as a function of (V_{max}). The subset of lower energy configurations where the particles are not overlapping, defined by ($d \geqslant 2r + \sigma_{LJ}$), shows a different trend to the full set which highlights the compromise between different energy levels.

as both V_{ref} and V_{net} exceed V_{max}, but remains sensitive when either value is lower. The value of (V_{max}) therefore determines the relative importance for different energy levels in the optimization process, which, as shown in Fig. 7, leads to a change in the fitting depending on the region.

This concludes the development of the network architecture; we now proceed to the evaluation of the method.

IV. COMPARISON WITH CONVENTIONAL COARSE-GRAINED MODEL

To demonstrate the ability to approximate interaction potentials with this type of neural network, several networks of different sizes are trained on datasets from a well-defined atomistic potential. The resulting networks are compared with a common type of coarse-grained potential, optimized with the same cost function.

A. Reference potential: CNTs

CNTs are a common type of nanoparticle composed entirely of carbon, that has attracted attention for many interesting properties, such as high stiffness, strength, and thermal and electrical conductivity. It has seen use in composites, both for its mechanical properties as reinforcement and for its other unique properties. For the purposes of this comparison, it provides a relatively simple rodlike geometry with straightforward options for conventional coarse graining (Fig. 9). Being nonpolar, interactions between CNTs can be modeled using the Lennard-Jones potential:

$$V_{\text{groupLJ}} = \sum_i \sum_j 4\varepsilon \left[\left(\frac{\sigma}{|\boldsymbol{d}_{ij}|} \right)^{12} - \left(\frac{\sigma}{|\boldsymbol{d}_{ij}|} \right)^{6} \right]. \quad (13)$$

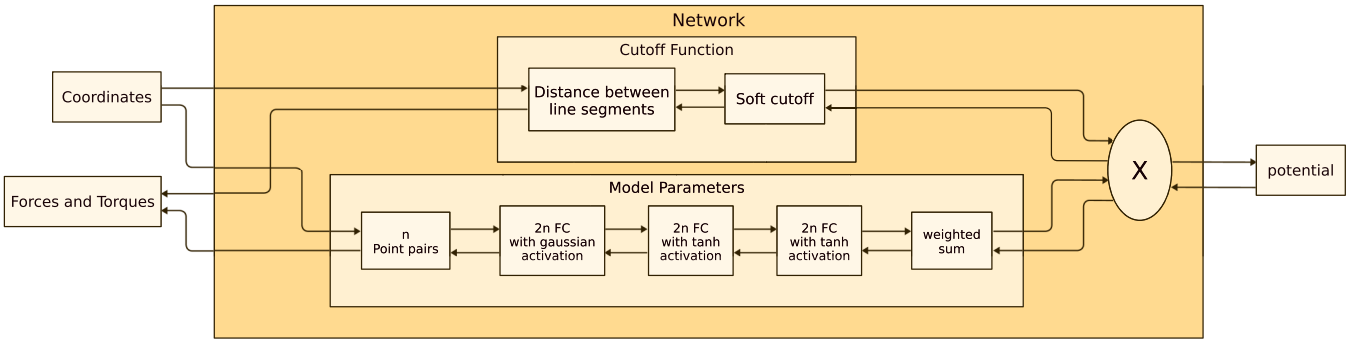


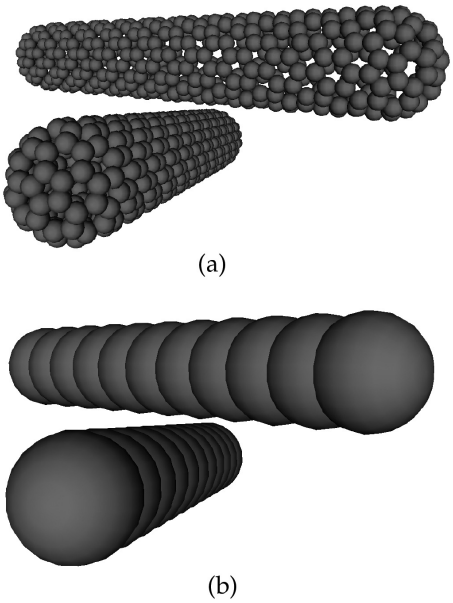
FIG. 8. Specific network configuration used in comparisons, with a specialization toward rodlike particles. The diagram includes all layers, parameterized by a single size variable n.

Interactions between groups of Lennard-Jones particles serve as an interesting benchmark of the model's performance, as they present a complex energy landscape with large gradients and singularities, making them a very challenging case for approximation with a smooth function.

Here, training data is generated by rejection sampling of the interaction space between two CNTs with a set maximum distance between the particles to remove irrelevant data points, as described in Appendix A 1.

B. Network configuration

The network used for these examples, seen in Fig. 8, consists of the geometric abstraction layer with a Gaussian activation for each pair, followed by two fully connected layers with tanh activation, followed by a weighted sum for the final potential. The cutoff uses the distance between line segments that have the same length as the CNTs without caps, with a cutoff distance of $2r_{\text{tube}} + 3\sigma_{\text{LJ}}$. The pair-distance smoothing parameter is set as $w = r_{\text{tube}}$.



(a) (b)

FIG. 9. Atomistic and coarse-grained models used as target and for comparison, respectively. (a) Two capped carbon nanotubes used for generating reference data. (b) Beads used for the coarse-grained representation.

V. COARSE-GRAINED REFERENCE

As a baseline for comparison with other methods, a coarse-grained model similar to the one presented by Chen *et al.* [15] is parameterized to replicate the same CNT potential. This potential models the CNT as a chain of beads, as shown in Fig. 9(b). For the purpose of comparison, particles are considered stiff, so only interparticle forces are considered. Each bead acts as a Lennard-Jones particle, and the parameters σ and ε are optimized by gradient descent with the same data and method as the network to minimize the cost function.

This model represents the simplest possible, and therefore most scalable, type of coarse-grained model for a rodlike particle [15]. The number of beads required to represent each particle scales linearly with the aspect ratio of the particle, and the parameters that can be changed are the bead spacing and bead−bead interaction potential. The smoothness of the potential along the particles length depends on both the bead spacing and the hardness and radius of the bead interaction potential. Assuming for example a hard sphere interaction between the beads, the bead spacing must be very short relative to the beads radius to approximate the interaction between hard cylindrical surfaces well. A softer bead potential like a Gaussian, on the other hand, would permit significantly longer bead spacing without introducing significant error due to the discretization. The Lennard-Jones interaction used here falls somewhere in between, having a relatively soft attractive component and a harder repulsive component. (a) Two capped carbon nanotubes used for generating reference data. (b) Beads used for the coarse-grained representation.

VI. NETWORK SIZE AND CONVERGENCE

The complexity of the potential to be approximated and network size will determine how good a fit is possible and how much training data is required to avoid overfitting, as seen in Fig. 10. Given enough training data, larger networks with more parameters can naturally achieve a better fit, but they are also more expensive to evaluate, which limits their utility for large-scale simulation. The optimal network size to use will therefore depend on both the amount of available training data and the required accuracy of the approximation.

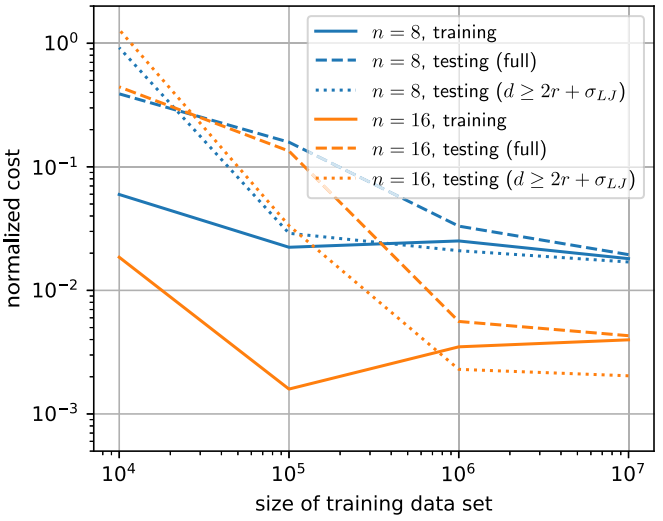


FIG. 10. Testing error as a function of the number of training samples used.

As seen in Fig. 11, the networks with $n > 4$ have a lower cost than the coarse-grained reference model for all datasets. While the fit improves for the full set of configurations, the greatest improvement is seen in the dataset that excludes overlapping configurations, where the $n = 16$ network shows a cost more than two orders of magnitude smaller than the coarse-grained model.

A. Error characteristics

While the average cost in the model is useful for monitoring convergence, its distribution in the parameter space is also of interest. Visualizing the full space of translations and orientations is infeasible, but two dimensional subspaces, where two coordinate components change, can be selected to give a useful representation for comparison, as seen in Figs. 12–15. These figures show both the potential and local cost for such a subspace relative coordinates, with plots split

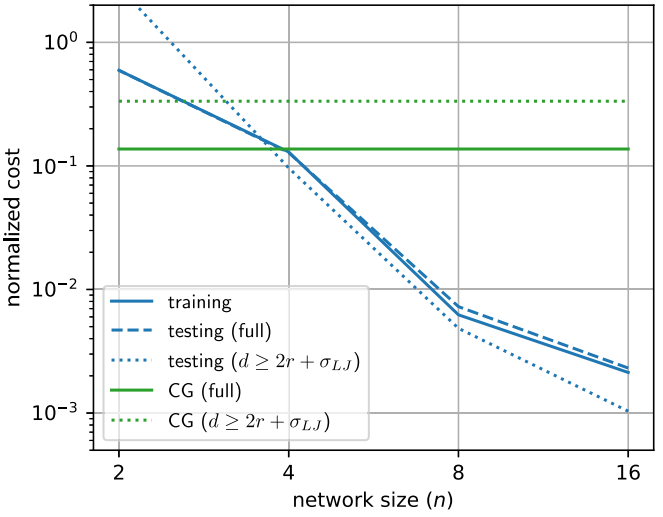


FIG. 11. Errors of networks with different sizes fitted to the potential, with 10^7 training samples.

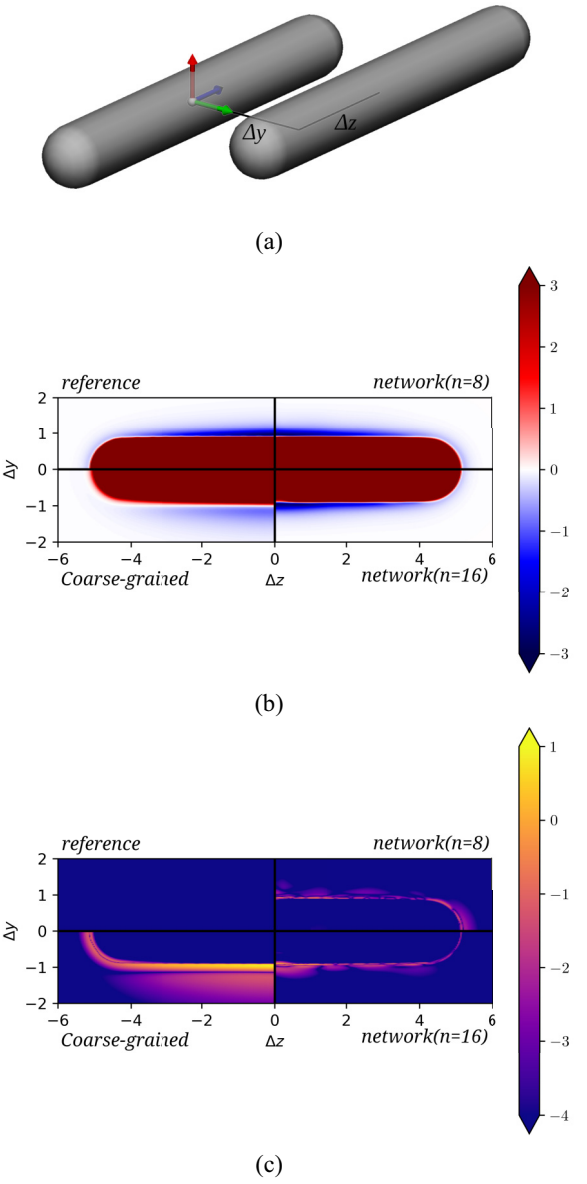


FIG. 12. Comparison for relative translation of parallel particles in the yz-plane. (a) Particle configuration, (b) Potential, (c) Local cost.

between the reference data, the coarse-grained model, and two different networks, to enable comparison. The local cost is simply the cost function applied to the reference potential and the approximation at a single point.

The high-energy regions in the middle are configurations where the particles overlap and are improbable to occur in an actual simulation, which means that a larger error is acceptable. This is accounted for by the cost function. It can clearly be observed that the networks achieve a better fit than the coarse-grained model in all the subspaces, with the networks showing relatively small cost outside of the high gradient regions, and overall looking very similar to the reference potential. The coarse-grained model is limited in its ability to fit due to the Lennard-Jones potential used for interaction between the beads and shows a relatively large error across much of the space in which interactions would normally take place. A more general spherical potential could likely achive

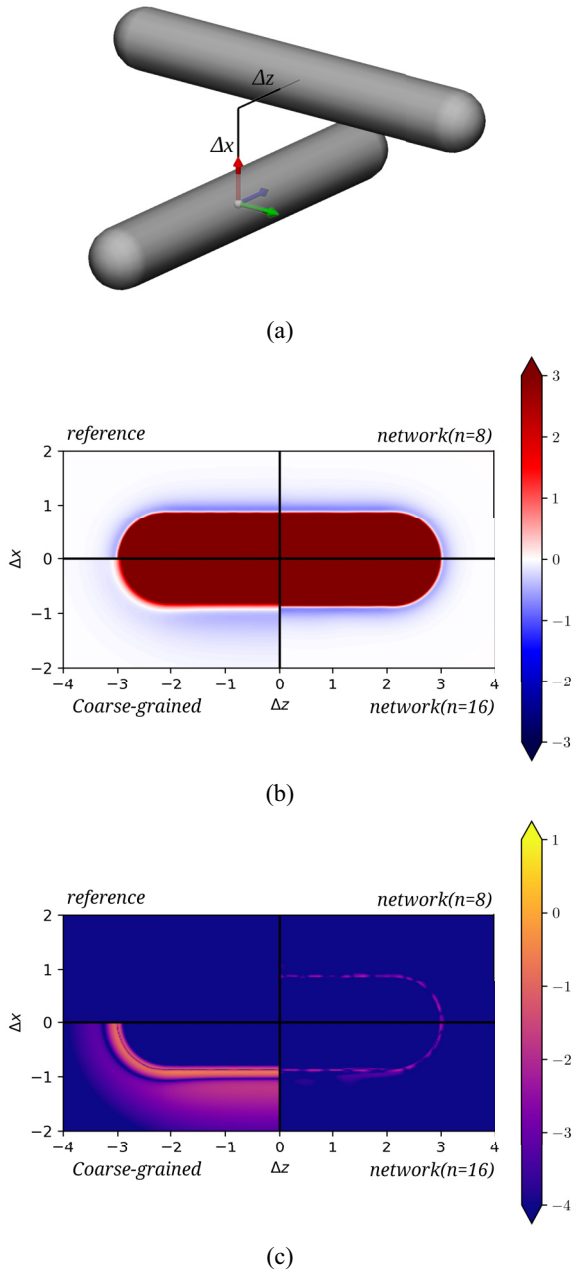


FIG. 13. Comparison for relative translation of perpendicular particles in the *yz*-plane. (a) Particle configuration, (b) Potential, (c) Local cost.

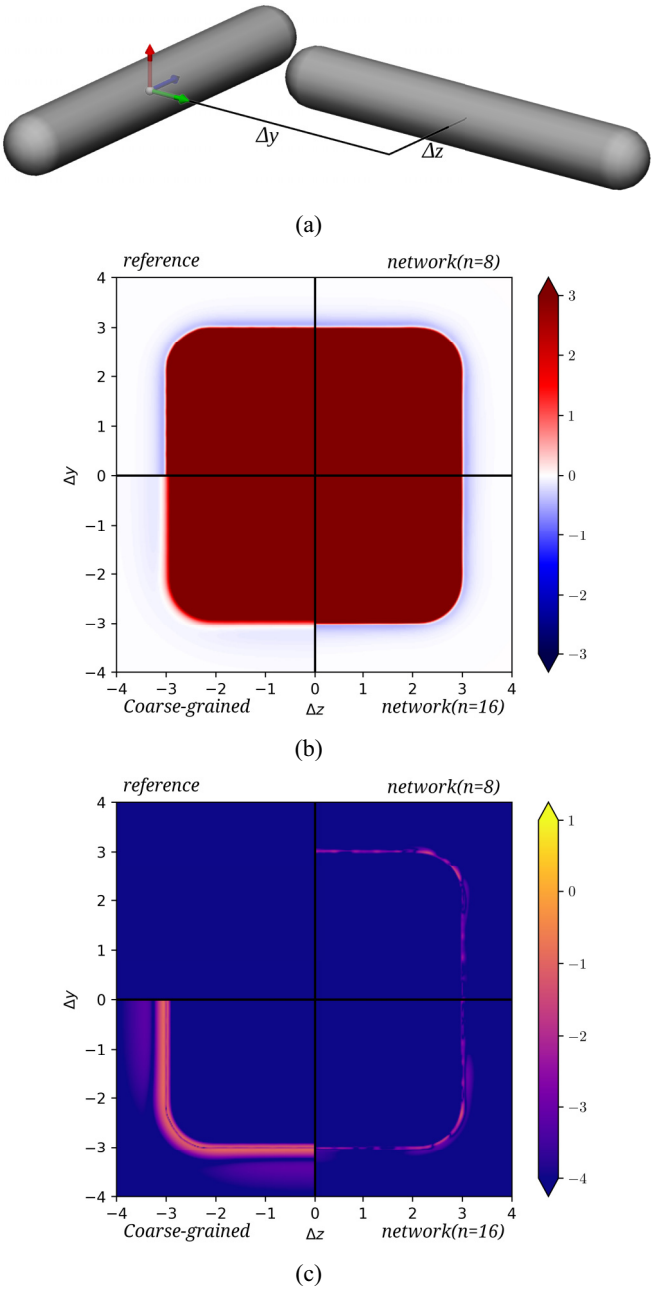


FIG. 14. Comparison for relative translation of perpendicular particles in the *xz*-plane. (a) Particle configuration, (b) Potential, (c) Local cost.

a slighly better fit, but would still be limited by the discretization of the particle into beads, which limits how large a gradient the potential can have without introducing roughness from the beads.

Predicting the actual effect of the error in the potential on a simulation is difficult, as the local error in individual points gives an incomplete picture of how well interaction behavior will be replicated. Particularly in the case of a reference potential with sharp gradients, small deviations in the geometry of the potential could give large errors around this gradient, without realistically affecting the simulation much. Similarly, the fitted function could be unable to fit to very sharp gradients. This would result in a smoother free-energy landscape for the approximation, which might have near-identical equilibrium distribution of particles to the exact potential, but different behavior in nonequilibrium simulations.

VII. SENSITIVITY TO NOISE IN TRAINING DATA

In some cases, such as when using molecular dynamics free-energy calculations as reference, computing precise values of the reference potential might be prohibitively expensive. If noisy data can be used to train the network, such models become much more viable. Training a model with noisy data will always result in some error, but depending on

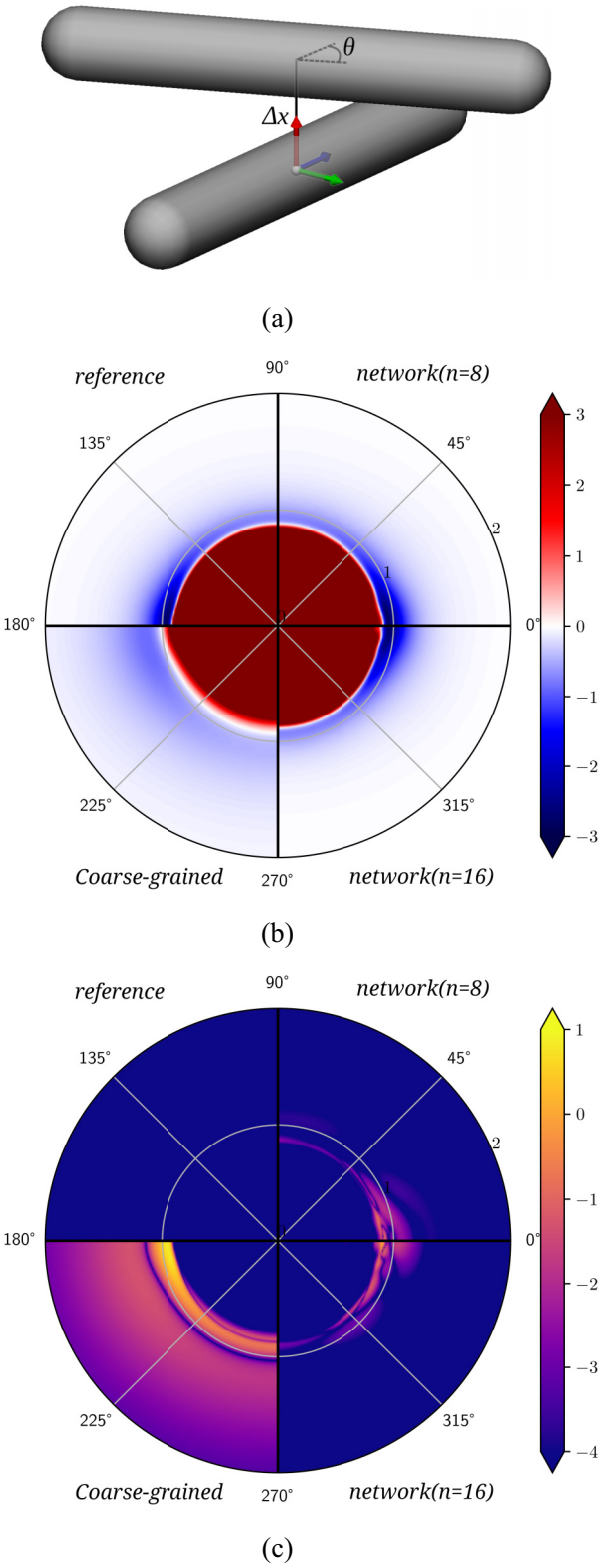


FIG. 15. Comparison for changing distance Δx and rotation θ. (a) Particle configuration, (b) Potential, (c) Local cost.

the models relative sensitivity to the noise and and the number of samples, it can be relatively small.

To test the sensitivity of the CNT model, different strengths of normally distributed noise were added to each sample in the training dataset, and networks were trained with the same procedure on each of the resulting noisy datasets. As seen in Fig. 16, the convergence for the test dataset and training dataset without the added noise is not significantly affected up to a noise level of $\sigma_{\text{noise}} = 0.25V_{\text{max}}$, although the fit for the lower energy region is affected somewhat earlier. It can also be seen that the network shows poor fitting to the noisy data it is being trained on, with the cost being very similar to the cost of the noise itself, shown in green. This demonstrates that the network can reject significant amounts of noise while still fitting to the underlying model. Since data from molecular dynamics simulation and Monte Carlo methods is inherently noisy, and reducing the noise level by running longer simulations or using more samples is expensive, the ability of an approximator to reject noise could prove very useful in the derivation of interaction models from data acquired with such methods.

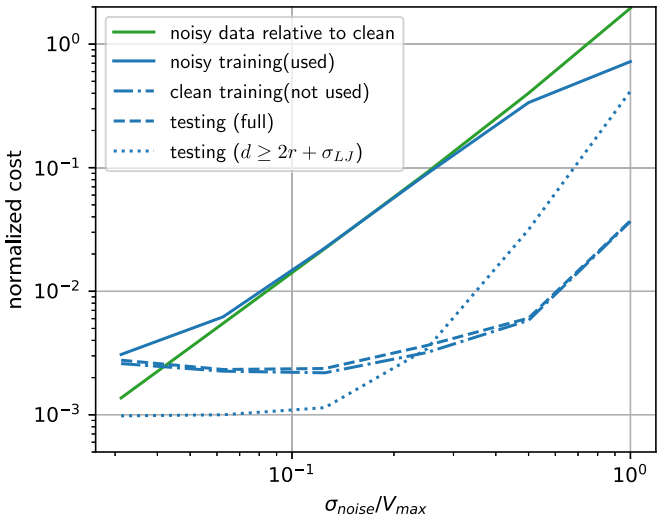


FIG. 16. Cost for different datasets for networks trained using noisy training data with 10^7 samples. The green line shows the cost of the noisy data used for training relative to the clean reference data and shows how noise is reflected in the cost.

VIII. IMPLEMENTATION AND PERFORMANCE

The usefulness of the model in any large-scale simulation ultimately depends on the cost of evaluation. While the actual performance will depend on both software implementation and the hardware used to run it, some scaling with the network size n is expected, and for very large networks, matrix multiplication should dominate the computation, leading to quadratic scaling with n, but for smaller networks, constant and linear terms may also be significant.

The proof-of-concept implementation is written in Futhark [16] and was inspired by the compositional neural networks by Tran *et al.* [17], which allows for relatively easy experimentation with complex network structures. The training uses a modified Adam [18] optimization algorithm with adaptive individual learning rates for all model parameters. For evaluation of a systems potential, interacting pairs of particles are found in parallel using cell lists implemented with an approach to irregular flattening shown by Elsman *et al.*

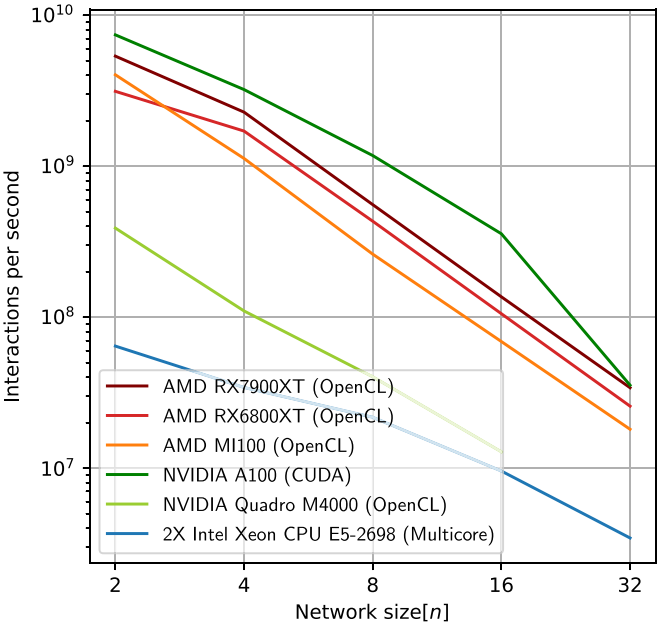


FIG. 17. Network evaluations per second for different devices. For the benchmark, the interaction evaluation is isolated by using pregenerated coordinates and not accumulating the results.

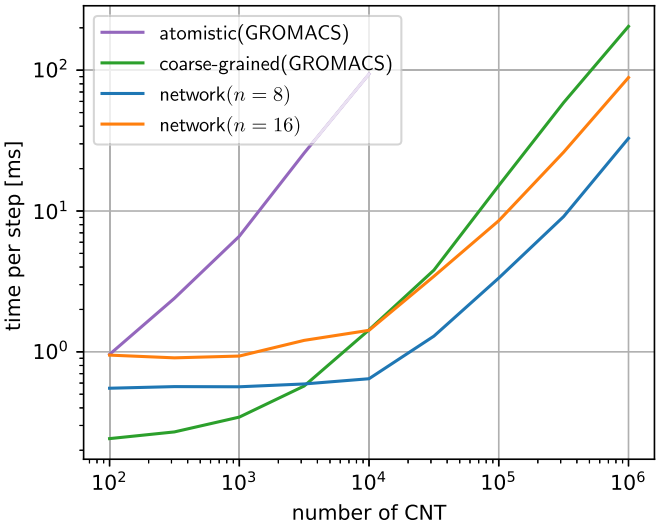


FIG. 18. Comparison of scaling relative to atomistic and coarse-grained systems in Gromacs, detailed in Appendix C.

[19], and forces and torques for each pair are aggregated as a generalized histogram [20]. The code can be compiled with multiple backends, including multicore CPU, OpenCL, HIP, and CUDA, for parallel execution on a variety of hardware. Figure 17 shows the performance of the implementation running on different platforms and devices. Due to the low number of parameters in the network, the whole network can fit in low-level cache, which limits memory bandwidth requirements, as only particle coordinates need to be loaded from shared memory. The implementation does not make use of the tensor-specific hardware present in many modern GPUs, and since significant parts of the network evaluation are matrix vector multiplications, this could likely be used to further improve performance. A comparison with existing methods in Fig. 18 shows the time to take a simulation step compared with a state-of-the-art implementation of existing methods for nonbonded interaction [21]. For larger systems, the proof-of-concept implementation is clearly faster than the coarse-grained approach, while simultaneously being much more accurate.

IX. DISCUSSION

A. Training with unknown potential

The reference potential used for demonstration purposes in this study is defined in such a way that getting an absolute potential at any given coordinate is possible without considering other coordinates. However, this is not the case for free-energy calculations in molecular dynamics. In methods like umbrella sampling, the potential is calculated locally and segments are joined together to give the global function. This is viable for low dimensional coordinate spaces where a dense sampling of the full space can be performed, but quickly becomes infeasible for higher dimensional coordinate spaces, like the one for rigid bodies. The gradient can be obtained at any point independently however, and assuming some smoothness of the function and the potential at a single point, given enough points, the gradient can in principle be used to define the potential.

In such a case, where only force and torque data are available, it is also in principle possible to train the network by differentiating the gradient calculation, with respect to the model parameters, to get the gradient for directly optimizing the force and torque components. Since the approximation is a smooth potential by definition, it should approach the real potential given the gradient at a sufficient number of points. This approach would enable direct use of molecular dynamics simulation data to the training of the network, but the number of data points required likely makes this infeasible for many cases.

B. Further generalization

While this study only considers potentials for rigid bodies, due to the geometric nature of the first layers point pairs, it could be further generalized by using more general coordinate transformations than rotation and translation. Given shape functions defining deformation, similar to those used in finite element modeling, additional displacement of the points could be defined, and conversely, the potential gradient on the points could be used in the calculation of deformations in the particle. With some modification, it might also be possible to define a potential network for polydisperse systems of particles where each particles length is used as an additional input in the network. Both of these extensions would likely significantly increase the required amount of training data required for an adequate fit, but could open up for new types of simulations.

C. Possible improvements

The optimal layer structure and activation functions remain an open question and are likely dependent on the type of

potential being interpolated. Considering the random distribution of the point pairs, a possible way to improve convergence is to start with a larger network containing more pairs, and during training, remove less effective pairs, again reducing the network size. This strategy would reduce sensitivity to the quality of the initial configuration.

The uniform random sampling in the relevant space of interactions used in this study is likely suboptimal given a limited number of training samples due to the apparent clustering seen in randomly sampled coordinates. It is therefore likely that the number of samples needed to ensure good generalization could be reduced by using a set of coordinates that fill the space more evenly. Low-discrepancy sequences such as the Sobol' sequence [22] are useful for this in Cartesian coordinates, but are not directly applicable to the combined space of relative positions and orientations. It is possible to generate a low-discrepancy sequence in $\mathbb{R}7$ and use rejection and projection to convert four of the components to a unit quaternion, but the effects of this process on the discrepancy and space-filling properties are not known.

Meaningfully quantifying the effect of error in the potential approximation on the properties of a simulated system is difficult, and further study is needed to improve understanding of this. In this study, high-energy configurations were given a lower weight in the optimization based on assumptions of their probability of occurrence being lower. This is reasonable given a sparse system in equilibrium, but for dense systems or systems subject to external forcing, these high-energy configurations become more relevant, and using a higher $V_{\max}$ would be necessary. It is also possible that a completely different cost function would be more suitable. When available, training data with a known gradient could also be used to improve the accuracy of forces and torques, which is more relevant when particles are forced externally.

D. Conclusion

The proposed network architecture demonstrates utility as a general purpose interpolant in the relative coordinate space of rigid bodies and achieves orders of magnitude better accuracy than typical coarse-grained methods of similar computational complexity for potentials that cannot feasibly be directly evaluated at large scales. Network training requires consideration of pretreatment of training data as well as network size parameters. The geometric first layer makes the optimal fit coordinate invariant and using back-propagation to calculate the force and torque ensures conservation of energy while reducing the required number of network parameters.

The method principally differs from a traditional coarse-grained approach in its sparse rather than dense interaction between points in the particles coordinate systems, and the more general, nonlinear combination of them. This is advantageous since the size and cost of evaluation of the model scales linearly with geometric complexity rather than quadratically and allows for fitting to more general functions.

The main benefit of this method is that realistic but computationally expensive interaction models can be constructed without the practical consideration of their direct application in a simulation. As long as the potential can feasibly be sampled to provide enough training data, around 10^7 times in this case, the trained network can reproduce the equivalent data, including forces and torques, in less than one second, with much higher accuracy than comparable conventional coarse-grained methods would provide. The ability to directly apply it to rigid bodies also reduces overhead from storing and integrating particle coordinates, and finding the interacting pairs. The removal of internal degrees of freedom can also drastically increase the length of time steps in simulation. This enables simulation with complex interactions on spatial and temporal scales that were previously infeasible and could help bridge the gap between our understanding of interactions at an atomistic level and observable macroscopic properties.

ACKNOWLEDGMENTS

This work was funded by the Swedish Research Council (Vetenskapsrådet) through the INTERFACE Project No. 2016-06119_VR. Parts of the computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council (Vetenskapsrådet) through Grant Agreement No. 2022-06725. We also gratefully acknowledge the use of resources from WWSC. We would like to thank Anna Broms and Anna-Karin Tornberg for many fruitful discussions, as well as Berk Hess for useful feedback. We would also like to thank Troels Henriksen, the primary author and maintainer of Futhark [16], for very quick and effective responses to issues with the compiler and assistance with benchmarking on multiple platforms.

APPENDIX A: TRAINING PROCEDURE

1. Coordinate sampling

To evenly sample data in the whole domain of interaction, rejection sampling is used. First, a plausible interaction coordinate with uniformly distributed relative position and orientation is generated and then rejected based on the cutoff function.

To generate a plausible interaction coordinate, a uniformly distributed R3 vector in the cube $(\pm1, \pm1, \pm1)$ is generated and then scaled to a box that includes all possible relative position where interaction could occur based on the particle dimensions and cutoff distance. To uniformly sample the orientation quaternion on the surface of unit hypersphere, uniformly distributed R4 vectors in the hypercube $(\pm1, \pm1, \pm1, \pm1)$ are generated and rejected until the vector falls withing the hypersphere, and then normalized to give the quaternion components.

2. Initialization

Since the point pairs represent the particle geometry, a good initialization of the network requires that the scale of the system is taken into account. Ideally, an initialized network should provide a usable cost gradient for the whole relevant interaction space, which will depend on particle geometry and the range of the interaction. The approach taken in this case

was to randomly place points in each particles coordinate system, such that they fall within a sphere encapsulating the particle.

The distribution of output values from this layer likewise depends on the size and geometry. For the activation in the next layer to be effective, the initial weights must be adjusted so the input to the activation function is distributed in a region with useful nonlinearity. Assuming sigmoid activation, this means the input cannot be too small, as it would fall in the approximately linear part of the function, effectively making the network incapable of nonlinear regression. It also cannot be too large, as most inputs are in regions where the function has almost no gradient, preventing the use of gradient descent optimization. Since the expected distances at initialization are proportional to the diameter sphere encapsulating the particle, its inverse can be used to normalize the initial weights.

APPENDIX B: IMPLEMENTATION DETAILS

1. Rod distance

The distance d_{rod} between two rods of length l, with relative distance $\mathbf{d}$ and orientation $\mathbf{q}$, can be expressed as the minimum distance between line segments, which is a quadratic optimization problem. With the line segments represented by the vectors

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ l/2 \end{bmatrix}, \qquad \mathbf{v} = \mathbf{q}\begin{bmatrix} 0 \\ 0 \\ l/2 \end{bmatrix}\mathbf{q}^{-1} \tag{B1}$$

the optimization problem can be written as

$$d_{\text{rod}} = \sqrt{\min\left(a\mathbf{u} + b\mathbf{v} + \mathbf{d}\right)^2 \begin{cases} -1 \leqslant a \leqslant 1 \\ -1 \leqslant b \leqslant 1 \end{cases}}. \tag{B2}$$

Since it is an optimization in only two variables, a and b, the problem can be solved by finding the global optimum, optima on the edges, and corners of the rectangular domain, filtering out of bound solutions, calculating distances, and then taking the minimum.

2. Compositional neural networks

When designing a neural network architecture, it is useful to split the network into different blocks that the data passes through. Compositional neural networks provide a framework for transferring these block schematics into functioning code. A network is defined as a set of the internal parameter and functions necessary to perform all optimization and evaluation operations.

The network type is parameterized by the type of the model parameter p, the type of input i, and the type of output o.

These networks can then be combined with functions that take the sets of functions and compose them into a new, valid set of functions. They can be chained, as seen in Fig. 19(a), so that data input is passed to the first network and its output of type m is passed as input to the second network. The networks

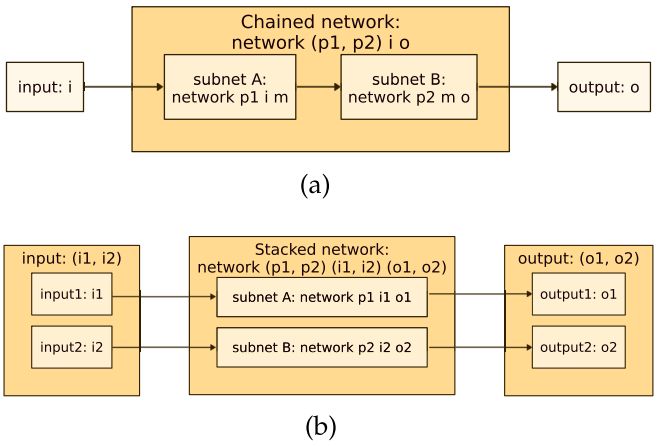


FIG. 19. Ways to compose smaller networks into bigger networks. (a) Chaining networks can be used to represent a path in a block diagram. (b) Stacking networks can be used to represent different paths in a block diagram.

can also be stacked, as seen in Fig. 19(b), so the input is split and each part is passed to one of the networks.

Through this kind of systematic combination of simpler networks that are easier to verify, complex networks can be created. Provided that the code of the subnetworks is correct, their combination will be as well.

3. Adaptive learning rate optimization

The optimization uses an adaptive learning rate η that adapts independently for each network parameter based on how correlated the gradient g is to the momentum m for this parameter. To quantify the correlation in a way that does not the depend on the magnitude of the quantities,

$$c = \frac{2gm}{g^2 + m^2} \tag{B3}$$

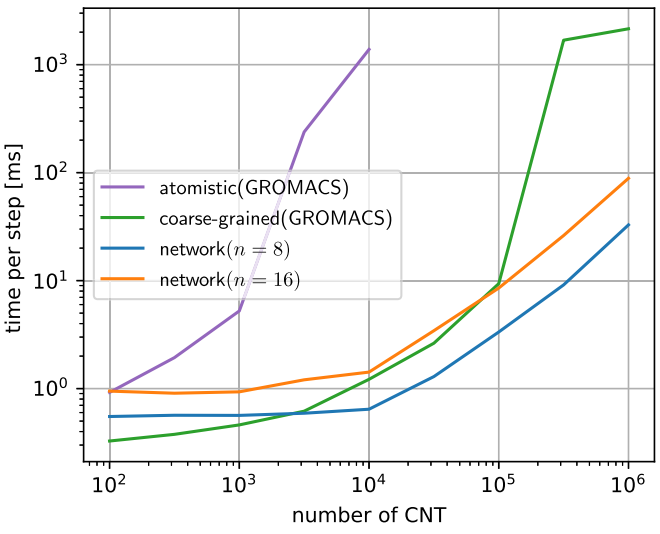


FIG. 20. Poor scaling behavior shown when doing updates on GPU.

is used. The learning rate is updated for each parameter by

$$\eta_{n+1} = \eta_n e^{\gamma c}. \tag{B4}$$

For parameters with positive correlations, the learning rate will increase, and for parameters with negative correlations, the learning rate will decrease. This is motivated by the idea of trying to get the maximum amount of information at each step of the optimization. The hyperparameter γ is set so that change happens at a low rate to avoid instability in case of spurious correlation between gradient and momentum.

APPENDIX C: GROMACS COMPARISON

To enable a reasonable comparison with a state-of-the-art implementation of atomistic and coarse-grained nonbonded interaction, equivalent systems of capped CNTs using the same particle coordinates were set up for each representation. The concentration of nanotubes was kept constant at 1.5625000×10^7 per μm^3, or approximately 2.5% by volume.

The simulations were set to only calculate nonbonded interactions and update particle positions, with no bonded interactions or constraints. To prevent the trajectories from diverging and maintain the same conditions, the particles were prevented from moving. In the case of the presented method, this was achieved by setting the time step to zero, and in Gromacs, this was done by freezing all atoms. Since particle coordinates are fixed, the neighbor list is constant; however, since this does not correspond to real simulation, the update interval was fixed to ten time steps for all systems.

The Gromacs systems used a spherical cutoff equal to the cutoff distance for the rod-specific cutoff used for the network.

The update for the Gromacs simulations using the standard leapfrog integration, but for the rigid body simulation, a simple viscous model approximating ellipsoids in a stokes fluid was used instead. This step is significantly more expensive per particle than simple rigid body dynamics, but is more in line with the intended use. All systems were run in the constant particle number, volume, and energy (NVE) ensemble to remove overhead and variance from thermostats and barostats.

Scaling issue in Gromacs

Prior to using freezing as a method to fix atoms in Gromacs, the two largest simulations of both types showed extremely poor performance, as seen in Fig. 20. Although the exact cause is not known, it likely involves the cache system on the RX 6800XT the simulations were run on, as the size required to store position, velocity, forces, and so on for all particle would fit in this cache for all other simulations. The behavior disappeared when freezing the atoms, which forced the update to take place on CPU, resulting in less information stored in the GPUs memory. While not indicative of Gromacs normal performance, this highlights an advantage of the reduced state of the rigid body representation; memory will be a limit for sufficiently large systems, and using a more compact representation results in better scaling.

[1] S. Shrestha, B. Wang, and P. Dutta, Nanoparticle processing: Understanding and controlling aggregation, Adv. Colloid Interface Sci. **279**, 102162 (2020).

[2] A. Rao, S. Roy, V. Jain, and P. P. Pillai, Nanoparticle self-assembly: From design principles to complex matter to functional materials, ACS Appl. Mater. Interfaces **15**, 25248 (2023).

[3] C. A. Silvera Batista, R. G. Larson, and N. A. Kotov, Nonadditivity of nanoparticle interactions, Science **350**, 1242477 (2015).

[4] B. Derjaguin and L. Landau, Theory of the stability of strongly charged lyophobic sols and of the adhesion of strongly charged particles in solutions of electrolytes, Prog. Surf. Sci. **43**, 30 (1993).

[5] E. J. W. Verwey, Theory of the stability of lyophobic colloids, J. Phys. Chem. **51**, 631 (1947).

[6] B. J. Berne and P. Pechukas, Gaussian model potentials for molecular interactions, J. Chem. Phys. **56**, 4213 (1972).

[7] E. de Miguel and E. Martín del Río, The isotropic–nematic transition in hard Gaussian overlap fluids, J. Chem. Phys. **115**, 9072 (2001).

[8] A. Gfrerrer, Rational interpolation on a hypersphere, Comput. Aided Geom. Des. **16**, 21 (1999).

[9] J. Behler and M. Parrinello, Generalized neural-network representation of high-dimensional potential-energy surfaces, Phys. Rev. Lett. **98**, 146401 (2007).

[10] S. 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).

[11] D. Dragoni, T. D. Daff, G. Csányi, and N. Marzari, Achieving DFT accuracy with a machine-learning interatomic potential: Thermomechanics and defects in bcc ferromagnetic iron, Phys. Rev. Mater. **2**, 013808 (2018).

[12] A. Kabylda, V. Vassilev-Galindo, S. Chmiela, I. Poltavsky, and A. Tkatchenko, Efficient interatomic descriptors for accurate machine learning force fields of extended molecules, Nat. Commun. **14**, 3562 (2023).

[13] R. Bhattoo, S. Ranu, and N. Krishnan, Learning the dynamics of particle-based systems with Lagrangian graph neural networks, Mach. Learn.: Sci. Technol. **4**, 015003 (2023).

[14] M. A. Nielsen, Neural networks and deep learning, 2018, http://neuralnetworksanddeeplearning.com/.

[15] H. Chen, L. Zhang, J. Chen, M. Becton, X. Wang, and H. Nie, Energy dissipation capability and impact response of carbon nanotube buckypaper: A coarse-grained molecular dynamics study, Carbon **103**, 242 (2016).

[16] T. Henriksen, N. G. W. Serup, M. Elsman, F. Henglein, and C. E. Oancea, Futhark: Purely functional gpu-programming with nested parallelism and in-place array updates, in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (ACM, New York, 2017), pp. 556–571.

[17] D. M. Tran, T. Henriksen, and M. Elsman, Compositional deep learning in futhark, in *Proceedings of the 8th ACM SIGPLAN International Workshop on Functional High-Performance and Numerical Computing* (ACM, New York, 2019), pp. 47–59.

[18] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[19] M. Elsman, T. Henriksen, and N. Gustav Westphal Serup, Data-parallel flattening by expansion, in *Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming* (ACM, New York, 2019), pp. 14–24.

[20] T. Henriksen, S. Hellfritzsch, P. Sadayappan, and C. Oancea, Compiling generalized histograms for gpu, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, Piscataway, NJ, 2020), pp. 1–14.

[21] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, Gromacs 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation, J. Chem. Theory Comput. **4**, 435 (2008).

[22] I. M. Sobol', On the distribution of points in a cube and the approximate evaluation of integrals, USSR Comput. Math. Math. Phys. **7**, 86 (1967).