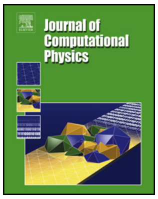
Journal of Computational Physics 473 (2023) 111692

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



Efficient long-range convolutions for point clouds

Yifan Peng[a,1], Lin Lin[b,c], Lexing Ying[d], Leonardo Zepeda-Núñez[e,*,2]

[a] Zhiyuan College, Shanghai Jiao Tong University, Shanghai 200240, China
[b] Department of Mathematics, University of California, Berkeley, United States of America
[c] Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States of America
[d] Department of Mathematics, Stanford University, Stanford, CA, United States of America
[e] Department of Mathematics, University of Wisconsin-Madison, Madison WI 53706, United States of America

ARTICLE INFO

Article history:
Received 25 February 2021
Received in revised form 16 September 2022
Accepted 10 October 2022
Available online 26 October 2022

Keywords:
Neural network
Long-range interactions
Non-uniform fast Fourier transform
Point cloud

ABSTRACT

The efficient treatment of long-range interactions (LRIs) for point clouds is a challenging problem in many scientific machine learning applications. To extract global information, one usually needs a large window size, a large number of layers, and/or a large number of channels. This can often significantly increase the computational cost. In this work, we present a novel neural network layer that directly incorporates long-range information for a point cloud. This layer, dubbed the *long-range convolutional* (LRC)-layer, leverages the convolutional theorem coupled with the non-uniform Fourier transform. In a nutshell, the LRC-layer mollifies the point cloud to an adequately sized regular grid, computes its Fourier transform, multiplies the result by a set of trainable Fourier multipliers, computes the inverse Fourier transform, and finally interpolates the result back to the point cloud. The resulting global all-to-all convolution operation can be performed in nearly-linear time asymptotically with respect to the number of input points. The LRC-layer is a particularly powerful tool when combined with local convolution as together they offer efficient and seamless treatment of both short- and long-range interactions. We showcase this framework by introducing a neural network architecture that combines LRC-layers with short-range convolutional layers to accurately learn the energy and force associated with a N-body potential. We also exploit the induced two-level decomposition and propose an efficient strategy to train the combined architecture with a reduced number of samples.

1. Introduction

Point cloud representations provide detailed information of objects and environments. The development of novel acquisition techniques, such as laser scanning, digital photogrammetry, light detection and ranging (LIDAR), 3D scanners, structure-from-motion (SFM), among others, has increased the interest of using point cloud representation in various applications such as digital preservation, surveying, autonomous driving [8], 3D gaming, robotics [38], and virtual reality [39]. In return, this new interest has fueled the development of machine learning frameworks that use point clouds as input. Historically, early methods used a preprocessing stage that extracted meticulously hand-crafted features from the point cloud,

---

[*] Corresponding author.
E-mail address: lzepedanunez@google.com (L. Zepeda-Núñez).
[1] Now at University of Chicago.
[2] Now at Google Research.

https://doi.org/10.1016/j.jcp.2022.111692

which were subsequently fed to a neural network [7,47,48,2], or they relied on voxelization of the geometry [49,52,43,36]. The PointNet architecture [41] was the first to handle raw point cloud data directly and learn features on the fly. This work has spawned several related approaches, aiming to attenuate drawbacks from the original methodology, such as PointNet++ [42], or to increase the accuracy and range of application [51,61,33,35].

Even though such methods have been quite successful for machine learning problems, they rely on an assumption of locality, which may produce large errors when the underlying task at hand exhibits long-range interactions (LRIs). To capture such interactions using standard convolutional layers, one can use wider window sizes, deeper networks, and/or a large number of features, which may increase the computational cost significantly. Several approaches have been proposed to efficiently capture such interactions in tasks such as semantic segmentation, of which the ideas we briefly summarize below. In the multi-scale type of approaches, features are progressively processed and merged. Within this family, there exist several variants, where the underlying neural networks can be either recursive neural networks [57], convolutional layers [53,54] or autoencoders [55,12]. Some works have proposed skip connections, following an U-net [44] type architecture [68,42], while others have focused on using a tree structure for the clustering of the points [30,59,19], or using an reference permutohedral lattices to compute convolutions [26] whose results are interpolated back to the point cloud [50]. Although these methods have been shown to be successful in a range of applications, when the task at hand presents symmetries, such as rotation, translation, and permutation invariance, there is no systematic framework to embed those symmetries into the algorithmic pipelines. Another line of work, relies on interpreting the point cloud as a graph and use spectral convolutions [6,11], whose cost can scale super-linearly when dealing with LRIs.

In applications of machine learning to scientific computing, several classical multilevel matrix factorizations have been rewritten in the context of machine learning [32], which have been adapted to handle LRIs in the context of end-to-end maps using voxelized geometries in [16,15,28,17] resulting in architectures similar to U-nets [44], which have been recently extended to point clouds in [34]. Note that due to underlying voxelization of the geometry, it may be difficult for these networks to generalize when the resolution of the voxelization changes.

The efficient treatment of LRI for point clouds is also a prominent problem in many physical applications such as molecular modeling and molecular dynamics simulation. While long-range electrostatic interactions are omnipresent, it has been found that effectively short-ranged models can already describe the N-body potential and the associated force field [4,64,65] for a wide range of physical systems. There have also been a number of recent works aiming at more general systems beyond this regime of effective short-range interactions, such as the work of Ceriotti and co-workers [22,23,37,45], as well as the works of [56,31,24,46,25,13,5,63]. The general strategy is to build parameterized LRIs into the kernel methods or neural network models, so that the resulting model can characterize both short-range, as well as long-range electrostatic interactions. In the neural network context, the computational cost of treating the LRIs using these methods can grow superlinearly with the system size.

The idea of this work is more closely aligned with the approaches in the molecular modeling community, which constructs a neural network layer to directly describe the LRI. In particular, we present a new *long-range convolutional* (LRC)-layer, which performs a global convolutional operation in nearly-linear time with respect to number of units in the layer. By leveraging the non-uniform Fourier transform (NUFFT) [14,20,3] technique, the LRC-layer implements a convolution with a point-wise multiplication in the frequency domain with trainable weights known as *Fourier multipliers*. The NUFFT is based on the regular fast Fourier transform (FFT) [10] with a fast gridding algorithms, to allow for fast convolution on unstructured data. This new LRC-layer provides a new set of descriptors that can seamlessly satisfy relevant symmetries. For instance, when the kernel of the LRI is rotationally invariant, such symmetry can be directly built into the parameterization of the Fourier kernel. Such descriptors can be used in tandem with the descriptors provided by short-range convolutional layers to improve the performance of the neural network.

Efficient training of a neural network with the LRC-layer for capturing the information of LRIs is another challenging problem. Short-range models can often be trained with data generated with a relatively small computational box (called the small-scale data), and they can be seamlessly deployed in large-scale systems without significantly increasing the generalization error. On the other hand, long-range models need to be trained directly with data generated in a large computational box (called the large-scale data), and the generation process of such large-scale data can be very expensive. For instance, in molecular modeling, the training data is often generated with highly accurate quantum mechanical methods, of which the cost can scale steeply as $\mathcal{O}(N^\alpha)$, where N is the system size and $\alpha \geq 3$. Therefore it is desirable to minimize the number of samples with a large system size. In many applications, the error of the effective short-range model is already modestly small. This motivates us to propose a *two-scale training strategy* as follows. We first generate many small-scale data (cheaply and possibly in parallel), and train the network without the LRC-layer. Then we use a small number of large-scale data, and perform training with both the short- and long-range convolutional layers.

In order to demonstrate the effectiveness of the LRC-layer and the two-scale training procedure, we apply our method to evaluate the energy and force associated with a model N-body potential that exhibit tunable short- and long-range interactions in one, two and three dimensions. The input point cloud consists of the atomic positions, and the output data include the N-body potential, local potential, and the force (derivative of the N-body potential with respect to atomic positions). In particular, the local potential and the force can be viewed as point clouds associated with the atomic positions. The evaluation of the N-body potential is a foundational component in molecular modeling, and LRI plays an important role in the description of ionic systems, macroscopically polarized interfaces, electrode surfaces, and many other problems in nanosciences [18]. Our result verifies that the computational cost of the long-range layer can be reduced from $\mathcal{O}(N^2)$ using a direct implementation, to $\mathcal{O}(N + N_{\texttt{FFT}} \log N_{\texttt{FFT}})$, where $N_{\texttt{FFT}}$ is the number of points of a reference grid. The value of $N_{\texttt{FFT}}$ depends on the separation between the particles in the point cloud. When the particles in the point cloud are approximately uniformly distributed, we have $N_{\texttt{FFT}} = \mathcal{O}(N)$. Furthermore, we demonstrate that the force, i.e. the derivatives of the potential with respect to *all* inputs can be also evaluated with $\mathcal{O}(N + N_{\texttt{FFT}} \log N_{\texttt{FFT}})$ cost. In terms of sample efficiency, we find that for the model problem under study here, the two-scale training strategy can effectively reduce the number of large-scale samples by over an order of magnitude to reach the target accuracy. This can be particularly valuable in the context of molecular modeling, where accurate data are often obtained from first principle electronic structure calculations. Such calculations are often very expensive for large scale systems, and the number of large-scale samples is thus limited.

This manuscript is organized as follows: Section 2 provides the algorithmic details of the LRC-layer. Section 3 provides the setting for the objective function that we seek to learn. Section 4 details of the architectures we benchmark, and we provide a complexity analysis of the LRC-layer showing the complexity reduction from quadratic to nearly-linear. Finally, Section 5 presents the numerical experiments showcasing the properties followed by Section 6, which provides the conclusions.

2. Long-range convolutional layer

Convolutional layers are perhaps the most important building-block in machine learning, due to their great success in image processing and computer vision. A convolutional layer convolves the input, usually an array, with a rectangular mask containing the trainable parameters. When the mask can be kept small (for example while extracting localized features), the convolution layer is highly efficient and effective. A different way for computing a convolution is to use the convolutional theorem as follows: (1) compute the Fourier transform of the input, (2) multiply with the Fourier transform of the mask, i.e., the Fourier multiplier, and (3) inverse Fourier transform back. In this case, the trainable parameters are the DOFs of the Fourier multipliers and the Fourier transforms are computed using the fast Fourier transform (FFT). This alternative approach is particularly attractive for smooth kernels with large support (i.e., smooth LRIs) because the computational cost does not increase with the size of the mask. To the best of our knowledge, this direction has not been explored for LRIs and below we detail now to apply this to point clouds.

Given a point cloud $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ and scalar weights $\{f_i\}_{i=1}^N$, we consider the problem of computing the quantity $u_i := \sum_{j=1}^N \phi_\theta(x_i - x_j) f_j$ at each x_i. Here the function $\phi_\theta(\cdot)$ is the kernel with a *generic* trainable parameter θ. At first glance the cost of this operation scales as $\mathcal{O}(N^2)$: we need to evaluate u_i for each point x_i, which requires $\mathcal{O}(N)$ work per evaluation. By introducing a generalized function $f(y) = \sum_i f_i \cdot \delta(y - x_i)$ and defining a function $u(x) = \int \phi_\theta(x-y) f(y) dy$, one notices that u_i is the value of $u(x)$ at $x = x_i$. The advantage of this viewpoint is that one can now invoke the connection between convolution and Fourier transform

$$\hat{u}(k) = \hat{\phi}_\theta(k) \cdot \hat{f}(k), \tag{1}$$

where $\hat{\phi}_\theta(k)$ is a trainable Fourier multiplier. This approach is suitable for point clouds since the trainable parameters are decoupled from the geometry of the point cloud. To make this approach practical, one needs to address two issues: (1) the non-uniform distribution of the point cloud and (2) how to represent the multiplier $\hat{\phi}_\theta(k)$.

2.1. Non-uniform distribution of the point cloud

Equation (1) suggests that one can compute the convolution directly using the convolution theorem, which typically relies on the FFT to obtain a low-complexity algorithm. Unfortunately, $\{x_i\}_{i=1}^N$ do not form a regular grid, thus FFT can not be directly used. We overcome this difficulty by invoking the NUFFT [14] (see Appendix B for further details), which serves as the corner-stone of our instance of the LRC-layer.[3]

The LRC-layer is summarized in Algorithm 1, where τ is chosen following [14]. The inputs of this layer are the point cloud $\{x_i\}_{i=1}^N$ and the corresponding weights $\{f_i\}_{i=1}^N$. The outputs are $u_i \equiv u(x_i)$ for $i = 1, \ldots, N$. The number of elements in the underlying grid $N_{\texttt{FFT}} = L_{\texttt{FFT}}^d$ is chosen such that the kernel is adequately sampled and the complexity remains low. As is will be shown in Section 5, one only needs a relatively small $L_{\texttt{FFT}}$. Even though the precise number is problem-specific, given that the goal is to approximate LRIs that are supposedly smooth, it can be captured with a relatively small number of Fourier modes.

The LRC-layer is composed of three steps as depicted in Fig. 1: (1) It computes the Fourier transform from the point cloud to a regular grid using the NUFFT algorithm (lines $2-5$ in Algorithm 1 and showcased in Fig. 2). (2) It multiplies the result by a set of trainable Fourier multipliers (line 6 in Algorithm 1). (3) It computes the inverse Fourier transform from the regular grid back to the point cloud (lines $7-9$ in Algorithm 1).

Within the LRC-layer in Algorithm 1, the only trainable component is the parameter θ of the Fourier multiplier $\hat{\phi}_\theta(k)$. The remaining components, including the mollifier $g_\tau(\cdot)$ and the Cartesian grid size, are taken to be fixed. One can, in

---

[3] We point out, that one could in practice use an fast summation algorithm, such as the fast multipole method (FMM) introduced by [21], to evaluate u_i. This would results in similar complexities if the kernel is fixed. However, the algorithm contains many different branches (e.g., if-statements) that depend on the kernel, which itself is trainable and therefore will evolve during the training stage. This would render the implementation much more cumbersome, particularly when computing derivatives.

Algorithm 1 Long-range convolutional layer.

```
Input: $\{x_i\}_{i=1}^N$, $\{f_i\}_{i=1}^N$
Output: $\{x_i\}_{i=1}^N$, $\{u_i\}_{i=1}^N$, where $u_i = \sum_{j=1}^N f_j \phi_\theta(x_i - x_j)$.
Define the generalized function: $f(x) = \sum_{j=1}^N f_j \delta(x - x_j)$
Mollify the Dirac deltas: $f_\tau(x) = \sum_{j=1}^N f_j g_\tau(x - x_j)$, where $g_\tau$ is defined in Appendix B
Sample in a regular grid: $f_\tau(x_\ell) = \sum_{j=1}^N g_\tau(x_\ell - x_j)$ for $x_\ell$ in grid of size $L_{\texttt{FFT}}$ in each dim
Compute FFT: $F_\tau(k) = \texttt{FFT}(f_\tau)(k)$
Re-scale the signal: $F(k) = \sqrt{\frac{\pi}{\tau}} e^{k^2 \tau} F_\tau(k)$
Multiply by Fourier multipliers: $\hat{v}(k) = \hat{\phi}_\theta(k) \cdot F(k)$
Re-scale the signal: $\hat{v}_{-\tau}(k) = \sqrt{\frac{\pi}{\tau}} e^{k^2 \tau} \hat{v}(k)$
Compute IFFT: $u_{-\tau}(x_\ell) = \texttt{IFFT}(\hat{v}_{-\tau})(x)$ for $x_\ell$ on the regular grid
Interpolate to the point cloud: $u_i = u(x_i) = u_{-\tau} * g_\tau(x_i)$
```

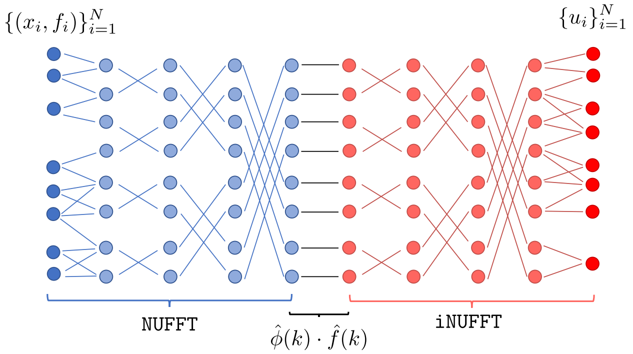


Fig. 1. Diagram of the LRC-layer. Starting from the cloud point $\{x_i\}_{i=1}^N$ with the scalar values $\{f_i\}_{i=1}^N$, we apply the NUFFT, to obtain the Fourier transform of f on a grid, we multiply it with the trainable Fourier multiplier, $\hat{\phi}$, and the use the inverse NUFFT to compute the values back to the point cloud.

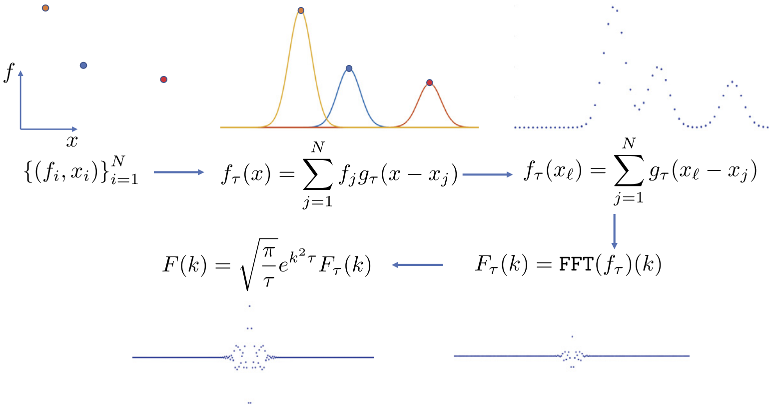


Fig. 2. Diagram of the NUFFT. Starting from the cloud point $\{x_i\}_{i=1}^N$, we form the mollified function f_τ, sample it in a regular grid, compute the Fourier transform $F_\tau(k)$ of the sampled function. Finally in order to obtain $F(k)$, we rescale the signal to undo the spatial convolution.

principle, train them as well, but it comes with a much higher cost. Among the steps of Algorithm 1, the sampling operator, the rescaling operator, the interpolation operator, and the Fourier transforms, are all *linear and non-trainable*. Therefore, derivative computations of backpropagation just go through them directly.

Algorithm 1 is presented in terms of only one single channel or feature dimension, i.e., $f_j \in \mathbb{R}$ and $u_i \in \mathbb{R}$. However, it can be easily generalized to multiple channels, for example $f_j \in \mathbb{R}^{d_1}$ and $u_i \in \mathbb{R}^{d_2}$. In this case, the Fourier multiplier $\hat{\phi}_\theta(k)$ at each point k is a $d_2 \times d_1$ matrix, and all Fourier transforms are applied component-wise.

2.2. Representation of the Fourier multiplier

A useful feature of the LRC-layer is that it is quite easy to impose symmetries on the Fourier multipliers. In fact, many recent research efforts have sought to incorporate symmetries in the neural network representation (see e.g., [62,60,65,58,9]). By restricting the model to satisfy these symmetries, the neural network model can often be more accurate and require fewer training data. Symmetry-preserving neural networks have found applications in quantum chemistry [40], image processing [47,48], molecular dynamics [66], among many other.

In the case of the LRC-layer, if the convolution kernel $\phi_\theta(\cdot)$ is constrained to have parity symmetry, rotational symmetry, smoothness or decay properties, these constraints can be imposed accordingly on the coefficients of the Fourier multipliers $\hat{\phi}_\theta(k)$. When the size of the training data is limited, it is often necessary to reduce the number of trainable parameters in order to regularize the kernel. For example, we may parameterize the Fourier multiplier as a linear combination of several predetermined functions on the Fourier grid. This is the procedure used in molecular modeling [22,56,31], and also in our numerical examples in equation (23). We also remark that the LRC-layer described here can be applied to point clouds a way similar to a standard convolution layer applied to images and multiple LRC-layers can be composed on top of each other.

3. Learning the *N*-body potential

To demonstrate the effectiveness of the LRC-layer, we consider the problem of learning the energy and force associated with a model N-body potential in the context of molecular modeling. As mentioned in Section 1, the potential evaluation often invokes expensive *ab-initio* calculations that one would like to bypass for efficiency reasons.

In general, the cost of evaluating the N-body potential scales quadratically with respect to the number of particles in the system. One usual technique to reduce the complexity is to leverage the nearsightedness principle, which postulates that the most relevant interactions tend to be localized, thus one can neglect the long-range interactions while only incurring on a negligible loss in accuracy. In several cases, however, long range interactions are needed to obtain the bulk properties of interest [67]. Due to the difficulty of tuning such systems, we use an analytical model which allows us to seamlessly transition form short- to long-range interactions.

The setup of this learning problem is as follows. First, we assume access to a *black-box* model potential, which consists of both short- and long-range interactions. However, we assume that the internal parameters of the potential are inaccessible to the training architecture and algorithm. A set of training samples are generated by the model, where each sample consists of a configuration of the points $\{x_i\}$ along with the potential and force. Second, we set up a deep neural network that includes (among other components) the LRC-layer for addressing the long-range interaction. This network is then trained with stochastic gradient type of algorithms using the collected dataset. The trained network can then be used for predicting the potential and forces for new point cloud configurations. These first component is described below, whereas the second component is described in detail in Section 4.

3.1. Black-box model problem

Model We suppose that $\Omega = [0, L]^d$, and we denote the point cloud by $\mathbf{x} = \{x_i\}_{i=1}^N \subset \Omega \subset \mathbb{R}^d$, for $d = 1, 2$, or 3. We define the total energy, the local potential and the forces acting on particle j by

$$U = \sum_{1\le i<j\le N} \psi(x_i - x_j), \qquad U_j(x) = \sum_{i\neq j} \psi(x_i - x), \qquad \text{and} \qquad F_j = -\partial_x U_j(x)|_{x=x_j}, \tag{2}$$

respectively, where the interaction kernel $\psi(r)$ is a smooth function, besides a possible singularity at the origin and decreases as $\|r\| \to \infty$. In this case we use two different kernels, the exponential and screened-Coulomb kernel, which are described in what follows.

Exponential kernel: Suppose Ω be the torus $[0, L]^d$ and that $\mathbf{x} = \{x_i\}_{i=1}^N \subset \Omega \subset \mathbb{R}^d$ for $d = 1, 2$, or 3. The exponential kernel is defined as

$$\psi^\mu(x - y) = e^{-\mu\|x-y\|}, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm over the torus. We define the total energy and the potential as

$$U = \sum_{i<j}^N e^{-\mu\|x_i - x_j\|} \qquad \text{and} \qquad U_j(x) = \sum_{i\neq j}^N e^{-\mu\|x_i - x\|}, \tag{4}$$

respectively. The forces are given by

$$F_j = -\partial_{x_j} U_j(x_j) = -\sum_{i\neq j}^N \frac{x_i - x_j}{\|x_i - x_j\|} \mu e^{-\mu\|x_i - x_j\|}. \tag{5}$$

Due to the exponential decay in the kernel only particles that are at $\mathcal{O}(\mu^{-1})$ distance will interact. For particles that are farther away we can effectively neglect them.

Screened-Coulomb kernel: In 3D, the screened-Coulomb potential with free space boundary condition is given by

$$\psi^\mu(x - y) = \frac{1}{4\pi\|x - y\|} e^{-\mu\|x-y\|}. \tag{6}$$

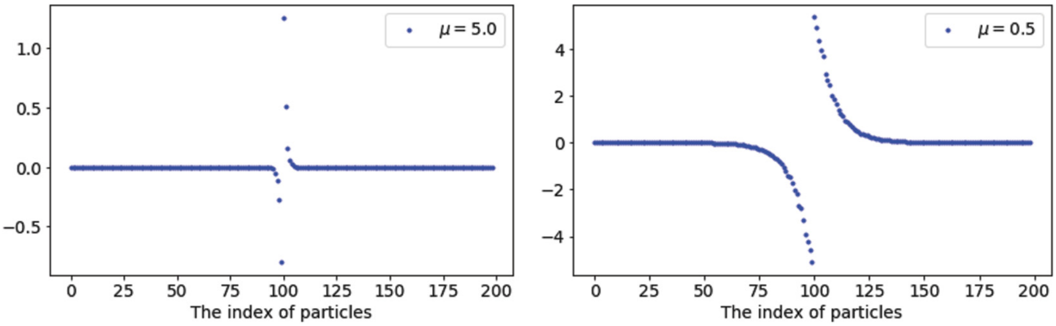


Fig. 3. The force contribution to particle x_{100} from other particles. Results are shown for two different characteristic interaction lengths.

Over the torus $[0, L]^d$, the kernel $\psi^\mu(x-y)$ is the Green's function $G^\mu(x, y)$ defined via

$$\Delta G^\mu(x, y) - \mu^2 G^\mu(x, y) = -\delta_y(x), \tag{7}$$

with the periodic boundary condition. In contrast with the exponential kernel, which is smooth up to a weak singularity at the origin, the screened-Coulomb potential exhibits an essential singularity at the origin. This type of singularities is found in many classical potentials [27], and it increases the impact of short-range interactions, while allowing the parameter μ to tune the LRIs. In order to compute the screened-Coulomb potential numerically, a spectral method is used: in particular,

$$\psi^\mu(x-y) = G^\mu(x, y) = \mathcal{F}^{-1}\left(\frac{e^{ik\cdot y}}{\|k\|^2 + \mu^2}\chi_\epsilon(k)\right), \tag{8}$$

where $\mathcal{F}^{-1}$ stands for the inverse Fourier transform and $\chi_\epsilon(k)$ is a smoothing factor, usually Gaussian, to numerically avoid the Gibbs phenomenon. Similar to the exponential case, the parameter μ controls the localization of the potential. In addition, the derivatives are taken numerically in the Fourier domain.

Visualization: To visualize the relation between μ and the characteristic interaction length in 1D, consider a given particle, e.g., x_{100} and compute the force contribution from the other particles. Fig. 3 shows that force contribution is extremely small outside a small interaction region for $\mu = 5.0$ while the interaction region for $\mu = 0.5$ is much larger.

Sampling We define a snapshot as one *configuration* of particles, $\mathbf{x}^\ell = \{x_j^{[\ell]}\}_{j=1}^N$, together with the global energy $U^{[\ell]}$ and the forces $F^{[\ell]}$, where ℓ is the index representing the number in the training/testing set. For simplicity, we suppose that the number of particles at each configuration is the same. We sample the configuration of particles $\mathbf{x}^\ell$ randomly, with the restriction that two particles can not be closer than a predetermined value $\delta_{\texttt{min}}$ in order to avoid the singularity. After an admissible configuration is computed we generate the energy and forces following the procedure above. This process is repeated until obtaining $N_{\texttt{sample}}$ snapshots.

4. Architecture

Our network architecture consists of separate descriptors for the short- interactions and long-range interactions, respectively. To capture the short-range interaction, we compute a local convolution using for each point only its neighboring points within a ball of predetermined radius. For the LRIs, we compute an all-to-all convolution using the LRC-layer introduced in Section 2, whose output is distributed to each particle and then fed to a sequence of subsequent layers. In this section we provide all relevant details corresponding to the architecture. We discuss the different building blocks, in particular, the descriptors, and how they are used to build the different networks. In addition, we provide details of how the derivatives can be taken in quasi-linear time in the case of the full-range network, and provide the complexity of the different operations.

4.1. Descriptors

Short-range descriptor For a given particle x_i, and an interaction radius R, we define $\mathcal{I}_i$, the interaction list of x_i, as the indices j such that $\|x_i - x_j\| < R$ where $\|\cdot\|$ stands for the distance over the torus $[0, L]^d$. I.e., $\mathcal{I}_i$ contains the indices of the particles that are inside a ball of radius R centered at x_i. Thus for each particle x_i we build the generalized coordinates $s_{i,j} = x_i - x_j$, and the short-range descriptor

$$\mathcal{D}^i_{\texttt{sr}} = \sum_{j\in\mathcal{I}_i} f_\theta(s_{i,j}), \tag{9}$$

where $f_\theta : \mathbb{R}^d \to \mathbb{R}^{m_{\texttt{sr}}}$ is a function represented by a neural network specified in what follows, where $m_{\texttt{sr}}$ is the number of short-range features. By construction $f_\theta(s)$ is smooth and it satisfies $f_\theta(s) = 0$ for $\|s\| > R$.

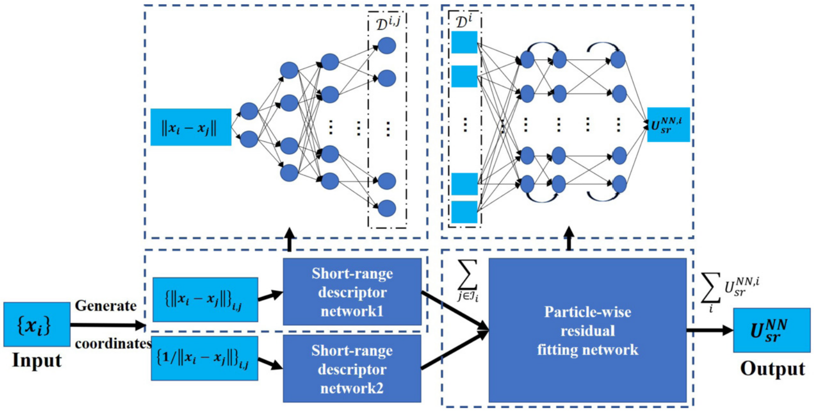


Fig. 4. The structure of short-range network for 1D case.

To simplify the discussion, we assume that there exists a maximal number of neighbors $N_{\texttt{maxNeigh}}$ for each x_i. We stack the neighbors in a tensor whose dimensions are constant across different particles. This value is chosen to be sufficiently large to cover the number of elements in the interaction list. If the cardinality of $\mathcal{I}_i$ is less than $N_{\texttt{maxNeigh}}$, we pad the tensor with dummy values.

In the 1D case the generalized coordinates are defined as

$$s_{i,j} = \|x_i - x_j\|, \qquad \text{and} \qquad r_{i,j} = \frac{1}{\|x_i - x_j\|} \tag{10}$$

for $j \in \mathcal{I}_i$.

To characterize f_θ in equation (9) we introduce two fully-connected neural networks $f_{\theta_1}, f_{\theta_2}$: $R^+ \to \mathbb{R}^{m_{\texttt{sr}}/2}$. Each network consists of five layers with the number of units doubling at each layer and ranging from 2 to 32. The activation function after each layer is tanh and the initialization follows Glorot normal distribution.

For particle x_i the short-range descriptor is defined as the concatenation of

$$\mathcal{D}^i_{1,\texttt{sr}} = \sum_{j\in\mathcal{I}_i} f_{\theta_1}(\hat{s}_{i,j})\hat{r}_{i,j} \qquad \text{and} \qquad \mathcal{D}^i_{2,\texttt{sr}} = \sum_{j\in\mathcal{I}_i} f_{\theta_2}(\hat{r}_{i,j})\hat{r}_{i,j}, \tag{11}$$

where $\hat{r}_{i,j}, \hat{s}_{i,j}$ are the normalized copies of $r_{i,j}$ and $s_{i,j}$ with mean zero and standard deviation equals to one. The mean and standard deviation are estimated by using a small number of snapshots. We multiply the network's output f_θ by $\hat{r}_{i,j}$ (which is zero if j is a dummy particle). This procedure enforces a zero output for particles not in the interaction list. The construction satisfies the condition $f_\theta(s) = 0$ for $\|s\| > R$.

In the short-range network, one concatenates the two descriptor above and feeds them particle-wise to the short-range fitting network. The fitting network $\mathcal{F}_{\texttt{sr}} : \mathbb{R}^{m_{\texttt{sr}}} \to \mathbb{R}$ is a residual neural network (ResNet) with six layers, each with 32 units. The activation function and initialization strategy are the same as the ones for the short-range descriptors. Fig. 4 shows the detailed architecture of the short-range network.

$$U^{\text{NN}}_{\texttt{sr}} = \sum_{i=1}^{N} \mathcal{F}(\mathcal{D}^i_{\texttt{sr}}) = \sum_{i=1}^{N} \mathcal{F}(\mathcal{D}^i_{1,\texttt{sr}}, \mathcal{D}^i_{2,\texttt{sr}}) \tag{12}$$

In 2D and 3D, there is a slight difference in the generalized coordinates: we compute

$$s_{i,j} = \frac{x_i - x_j}{\|x_i - x_j\|} \qquad \text{and} \qquad r_{i,j} = \frac{1}{\|x_i - x_j\|}, \tag{13}$$

where $s_{i,j}$ is a vector now. The local descriptors are defined in the following forms:

$$\mathcal{D}^i_{1,\texttt{sr}} = \sum_{j\in\mathcal{I}_i} f_{\theta_1}(s_{i,j})\hat{r}_{i,j} \qquad \text{and} \qquad \mathcal{D}^i_{2,\texttt{sr}} = \sum_{j\in\mathcal{I}_i} f_{\theta_2}(\hat{r}_{i,j})\hat{r}_{i,j} \tag{14}$$

Long-range descriptor We feed the LRC-layer with the raw point cloud represented by $\{x_i\}_{i=1}^N$ with weights $\{f_i\}_{i=1}^N$, which for simplicity can be assumed to be equal to one here, i.e., $f_i = 1$ for $i = 1, ..., N$. The output of the layer is a two-dimensional tensor $u^k(x_i)$ with $i = 1, \ldots, N$ and $k = 1, \ldots, K_{\texttt{chnls}}$. Then for each x_i, its corresponding slice given by the vector $[u^1(x_i), u^2(x_i), \cdots, u^{K_{\texttt{chnls}}}(x_i)]$, is fed to a function $g_\theta : \mathbb{R}^{K_{\texttt{chnls}}} \to \mathbb{R}^{m_{\texttt{lr}}}$, which is represented by a neural network

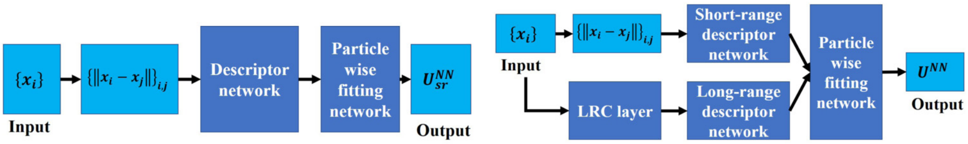


Fig. 5. (left) The short-range network architecture. (right) The full-range network architecture.

with non-linear activation functions. Here θ is a generic set of trainable parameters and $m_{\mathtt{lr}}$ is the number of long-range features. The descriptor for particle x_i, which depends on all the other particles thanks to the LRC-layer, is defined by

$$\mathcal{D}^i_{\mathtt{lr}} = g_\theta(u^1(x_i), u^2(x_i), \cdots, u^{K_{\mathtt{chnls}}}(x_i)) \tag{15}$$

4.2. Networks

Short-range Network When only the short-range interaction is present, the short-range descriptor for each particle is fed particle-wise to a fitting network $\mathcal{F}_{\mathtt{sr}} : \mathbb{R}^{m_{\mathtt{sr}}} \to \mathbb{R}$. In this case $\mathcal{F}_{\mathtt{sr}}(\mathcal{D}^i_{\mathtt{sr}})$ only depends on particle x_i and its neighbors. Finally, the contributions from each particle are accumulated so the short-range neural network (NN) energy and forces are given by

$$U^{\mathrm{NN}}_{\mathtt{sr}} = \sum_{i=1}^{N} \mathcal{F}_{\mathtt{sr}}(\mathcal{D}^i_{\mathtt{sr}}) \qquad \text{and} \qquad \left(F^{\mathrm{NN}}_{\mathtt{sr}}\right)_j = -\partial_{x_j} U^{\mathrm{NN}}_{\mathtt{sr}} \tag{16}$$

respectively (see Fig. 5(left)). The derivatives are computed using Tensorflow [1] directly. This network as shown by [65] is rotation, translation, and permutation invariant [58].

We point out that this architecture can be understood as a non-linear local convolution: for each particle i one applies the same function f_θ to each of its neighbors. The result is then pooled into the descriptor $\mathcal{D}^i_{\mathtt{sr}}$, then processed locally by $\mathcal{F}_{\mathtt{sr}}$ (akin to a non-linear convolution with a filter of width one), and finally pooled globally into $U^{\mathrm{NN}}_{\mathtt{sr}}$.

Full-range network When both the short-range and long-range interactions are present, the long range descriptor and the local descriptor are combined and fed particle-wise to a fitting network $\mathcal{F} : \mathbb{R}^{m_{\mathtt{sr}}+m_{\mathtt{lr}}} \to \mathbb{R}$ to produce the overall neural network (NN) energy and forces

$$U^{\mathrm{NN}} = \sum_{i=1}^{N} \mathcal{F}(\mathcal{D}^i_{\mathtt{sr}}, \mathcal{D}^i_{\mathtt{lr}}), \qquad \text{and} \qquad \left(F^{\mathrm{NN}}\right)_j = -\partial_{x_j} U^{\mathrm{NN}} \tag{17}$$

respectively (see Fig. 5(right)). Following Section 2, the long-range descriptor is translation invariant by design and can be easily made rotation invariant. Furthermore, it is well known [58] that this construction is permutation invariant. From the structures shown in Fig. 5 (we provide more detailed schematics in Fig. 4 and Fig. 6), it is clear that we can recover the first architecture from the second, by zeroing some entries at the fitting network, and removing the LRC-layer.

4.3. Derivatives

For the computation of the forces in equation (16) one needs to compute the derivatives of the total energy U^{NN} with respect to the inputs, in nearly-linear time. The main obstacle is how to compute the derivatives of the LRC-layer with respect to the point cloud efficiently. To simplify the notation, we only discuss the one-dimensional case, i.e., $d = 1$, but the argument can be seamlessly extended to higher dimensional cases, $d > 1$.

Recall that $u_i = \sum_{j=1}^{N} \phi_\theta(x_i - x_j) f_j$, then the Jacobian of the vector u with respect to the inputs is given by

$$(\nabla u)_{i,j} := \frac{\partial u_i}{\partial x_j} = \begin{cases} -f_j \phi'_\theta(x_i - x_j), & \text{if } j \neq i, \\ \sum_{k \neq i} f_k \phi'_\theta(x_i - x_k), & \text{if } j = i. \end{cases} \tag{18}$$

As it will be explained in the sequel, for the computation of the forces in equation (16) one needs to compute the application of the Jacobian of u to a vector. For a fixed vector $v \in \mathbb{R}^N$, the product $(\nabla u) \cdot v$ can be written component-wise as

$$\begin{aligned} ((\nabla u) \cdot v)_i &= -\sum_{j \neq i} v_j f_j \phi'_\theta(x_i - x_j) + v_i \sum_{j \neq i} f_j \phi'_\theta(x_i - x_j), \\ &= -\sum_{j=1}^{N} v_j f_j \phi'_\theta(x_i - x_j) + v_i \sum_{j=1}^{N} f_j \phi'_\theta(x_i - x_j), \end{aligned}$$

where we have added $\pm v_i f_i \phi'(0)$ in the last equation and then distributed it within both sums. Let us define the following two long-range convolutions

$$w_i = -\sum_{j=1}^{N} v_j f_j \phi'_\theta(x_i - x_j), \qquad \text{and} \qquad p_i = \sum_{j=1}^{N} f_j \phi'_\theta(x_i - x_j), \tag{19}$$

each of which can be performed in $\mathcal{O}(N + N_{\mathtt{FFT}} \log N_{\mathtt{FFT}})$ steps using the NUFFT algorithm combined with the convolution theorem. In this case the derivative of ϕ can be computed numerically in the Fourier domain to a very high accuracy. Now one can leverage the expression above to rewrite $(\nabla u) \cdot v$ as

$$((\nabla u) \cdot v)_i = w_i + v_i p_i, \tag{20}$$

which can then be computed in nearly-linear time. The same is also true for $v \cdot (\nabla u)$.

4.4. Complexity

It is clear that the energy can be evaluated in nearly-linear complexity. In what follows we show that the force computation is also of nearly-linear. For simplicity we focus on the one-dimensional network and assume that $K_{\mathtt{chnls}} = 1$, $\mathcal{O}(m_{\mathtt{sr}}) = \mathcal{O}(m_{\mathtt{lr}}) = \mathcal{O}(1)$ and that the depth of the neural networks is $\mathcal{O}(1)$. As defined in the prequel the forces are given by $F^{\mathrm{NN}} = -\nabla_x U^{\mathrm{NN}}$, which can be written component wise as

$$F^{\mathrm{NN}}_j = -\partial_{x_j} U^{\mathrm{NN}} = -\sum_{i=1}^{N} \left[\partial_1 \mathcal{F}(\mathcal{D}^i_{\mathtt{sr}}, \mathcal{D}^i_{\mathtt{lr}}) \partial_{x_j} \mathcal{D}^i_{\mathtt{sr}} + \partial_2 \mathcal{F}(\mathcal{D}^i_{\mathtt{sr}}, \mathcal{D}^i_{\mathtt{lr}}) g'_\theta(u_i) \partial_{x_j} u_i \right], \tag{21}$$

or in a more compact fashion as

$$F^{\mathrm{NN}} = -\nabla U^{\mathrm{NN}} = -\left(v_{\mathtt{sr}} \cdot D_{\mathtt{sr}} + v_{\mathtt{lr}} \cdot \nabla u \right). \tag{22}$$

Here $v_{\mathtt{sr}}$, and $v_{\mathtt{lr}}$ are vectors defined component-wise as $(v_{\mathtt{sr}})_i = \partial_1 \mathcal{F}(\mathcal{D}^i_{\mathtt{sr}}, \mathcal{D}^i_{\mathtt{lr}})$, and $(v_{\mathtt{lr}})_i = \partial_2 \mathcal{F}(\mathcal{D}^i_{\mathtt{sr}}, \mathcal{D}^i_{\mathtt{lr}}) g'_\theta(u_i)$. In addition $(D_{\mathtt{sr}})_{i,j} = \partial_{x_j} \mathcal{D}^i_{\mathtt{sr}}$ and ∇u is defined above.

The first term in the right-hand side is easy to compute, given that $D_{\mathtt{sr}}$ is sparse: the i, j entry is non-zero only if the particle x_i is in the interaction list of x_j. Given that the cardinality of the interaction list is bounded, $D_{\mathtt{sr}}$ has $\mathcal{O}(N)$ non-zero entries in which each entry requires $\mathcal{O}(1)$ work, thus the first term in the right-hand side of equation (22) can be computed in $\mathcal{O}(N)$. At first glance the complexity of second term seems to be much higher. However, as discussed above, by using equation (20), we can apply the matrix (or its transpose) to a vector in $\mathcal{O}(N + N_{\mathtt{FFT}} \log N_{\mathtt{FFT}})$ time and the computation of vector $v_{\mathtt{lr}}$ requires $\mathcal{O}(1)$ work per entry, thus resulting in a complexity of $\mathcal{O}(N + N_{\mathtt{FFT}} \log N_{\mathtt{FFT}})$ for computing the second term in equation (22). Finally, adding both contributions together results in an overall $\mathcal{O}(N + N_{\mathtt{FFT}} \log N_{\mathtt{FFT}})$ complexity for the forces.

To summarize, both the computation of the energy and the forces can be performed in $\mathcal{O}(N)$ time provided that the particles are not highly clustered, i.e., the pair-wise distance between particles is bounded from below by a constant. This stems from the expression for $L_{\mathtt{FFT}}$, which depends on the separation of the particles in the point cloud [14]. In addition, we point out that the bottleneck on the NUFFT, the spread of the Dirac deltas to the Fourier grid by mollification. For each particle one needs to perform a search of the nearby points and then compute distances and then evaluate the mollification kernel. This is a very memory-intensive operation, which is often order of magnitude slower than floating point operations in modern CPUs.

5. Numerical experiments

The loss function is the mean squared error of the forces

$$\frac{1}{N_{\mathtt{sample}}} \sum_{\ell=1}^{N_{\mathtt{sample}}} \sum_{i=1}^{N} \| F^{\mathrm{NN}}_\theta(x^{[\ell]}_i) - F^{[\ell]}_i \|^2,$$

where the i-index runs on the points of each snapshot, and ℓ runs on the test samples. We also generate 100 snapshots of data to test the performance of network. This particular loss could lead to shift the potential energy by up to a global constant, which can be subsequently fixed by including the error of the energy in the loss [65]. For the testing stage of we use the relative ℓ^2 error of the forces as metric, which is defined as

$$\epsilon_{\mathtt{rel}} := \sqrt{\frac{\sum_{\ell,i} \|F^{[\ell]}_i - F^{\mathrm{NN}}_\theta(x^{[\ell]}_i)\|^2}{\sum_{\ell,i} \|F^{[\ell]}_i\|^2}}.$$

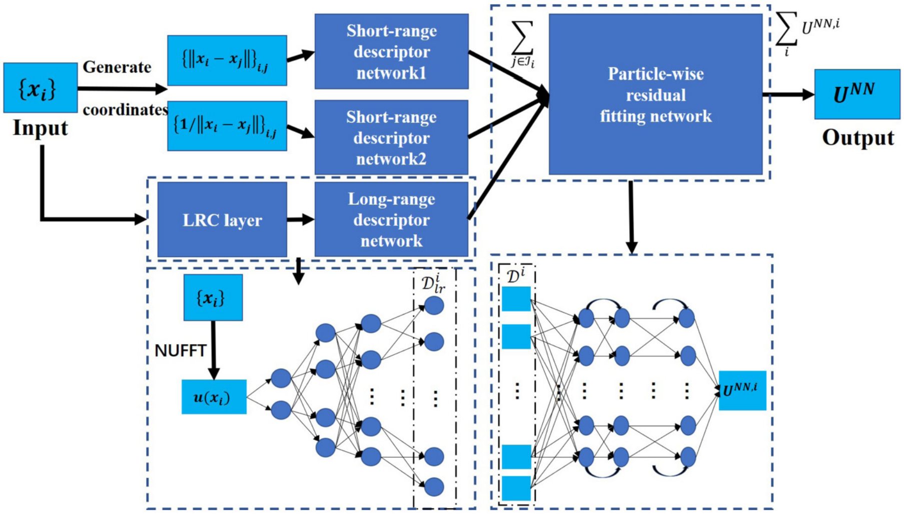


Fig. 6. The structure of full-range network.

The training parameters are listed below.

The experiments shown in the sequel are designed to provide a fair comparison with state-of-the-art methods for localized interactions. They showcase that, by adding a single LRC-layer, one can outperform these methods significantly.

The kernels ψ used in the experiment typically exhibit two interaction lengths: $\psi(\cdot) \equiv \alpha_1 \psi^{\mu_1}(\cdot) + \alpha_2 \psi^{\mu_2}(\cdot)$, where each of ψ^{μ_1} and ψ^{μ_2} is either a simple exponential kernel or screened-Coulomb kernel (also known as the Yukawa kernel). For each of ψ^{μ_1} and ψ^{μ_2}, the superscripts denote the reciprocal of the interaction length, i.e., length scale $\sim \mu_1^{-1}$ or $\sim \mu_2^{-1}$. Without loss of generality, $\mu_1 > \mu_2$, so that μ_1 corresponds to the short-range scale and μ_2 the long-range scale. We also assume that $0 \le \alpha_2 \le \alpha_1$ and $\alpha_1 + \alpha_2 = 1$, so that the effect of the long-range interaction can be smaller in magnitude compared to that of the short-range interaction. This assumption reflects the fact that in many cases, the short-range interactions are dominant, which in return has been used by recent ML-based methods to neglect the long-range interactions with only a marginal loss of accuracy (see, e.g., [4,64]). In the special case of $\alpha_2 = 0$, the kernel exhibits only a single scale $\sim \mu_1^{-1}$. The precise definition of the kernel depends on the spatial dimension and boundary conditions, which are explained in Section 3.

For a fixed set of kernel parameters $(\mu_1, \mu_2, \alpha_1, \alpha_2)$, we consider two types of data: large- and small-scale data, generated in the domains $\Omega_{\mathtt{lr}}$ and $\Omega_{\mathtt{sr}}$ respectively (details to be defined in each experiment).

The Fourier multiplier within the LRC-layer is parameterized as

$$\hat{\phi}_{\beta,\lambda}(k) = \frac{4\pi\beta}{|k|^2 + \lambda^2}, \tag{23}$$

where β and λ are trainable parameters. This is a simple parameterization, and a more complex model can be used as well with minimal changes to the procedure. For all experiments shown below, two kernel channels are used and as a result there are only four trainable parameters in the LRC-layer.

The numerical results aim to show namely two properties: i) the LRC-layer is able to efficiently capture LRIs, ii) the two-scale training strategy can reduce the amount of large-scale data significantly, and iii) the accuracy of the LRC-layer depends weakly on the number of Fourier modes, provided that the multiplier is properly sampled. To demonstrate the first property, we gradually increase the interaction length of the kernel. The accuracy of the short-range network with a fixed interaction radius is supposed to decrease rapidly, while using the LRC-layer improves the accuracy significantly. To show the second property, we generate data with two interaction lengths and train the full-range network using the one- and two-scale strategies. To show the third property we generate data with two interactions lengths and we train similar networks but with different values of $N_{\mathtt{FFT}}$. Finally, we also aim to demonstrate that the LRC-layer is competitive against a direct convolution in which the all-to-all computation is performed explicitly.

For the training procedure we use the Adam optimizer [29] along with an exponential scheduler. The learning rate with the initial learning rate taken to be 0.001 and, for every 10 epochs, it decreases by a factor of 0.95. In order to balance the computational time and the accuracy, a multi-stage training is adopted, where at each stage we modify the batch-size and the number of epochs. In particular, four stages are used: we start using a batch size of 8 snapshots and train the network 200 epochs and then at each stage we double both the size of the batch size and the number of epochs. In the two-scale training strategy, the same training parameters defined above are used for each stage.

Table 1
Relative testing error for trained screened-Coulomb type 1D models with $\alpha_1 = 1, \alpha_2 = 0$, and varying μ_1. Notice that μ_2 can be arbitrary here given that $\alpha_2 = 0$.

μ_1	0.5	1.0	2.0	5.0	10.0
short-range network	0.05119	0.02919	0.00597	0.00079	**0.00032**
full-range network	**0.00828**	**0.00602**	**0.00336**	**0.00077**	0.00054

Table 2
Error with respect to $N_{\mathtt{FFT}}$ in the 1D case.

$N_{\mathtt{FFT}}$	None	63	125	251	501
Relative testing error	0.06143	0.00536	0.00546	0.00545	0.00539

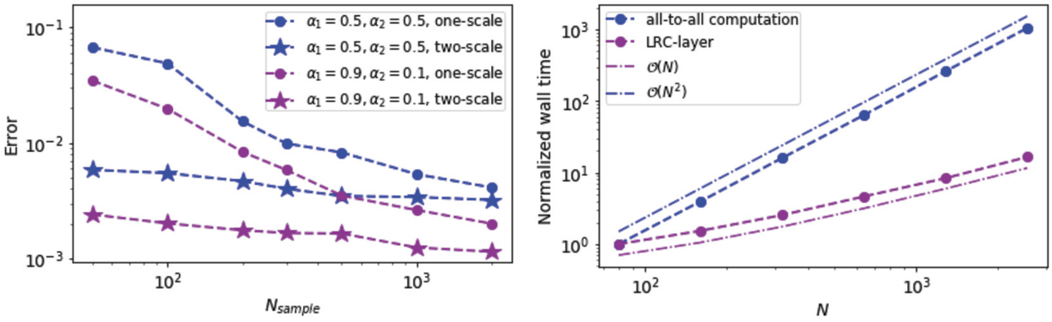


Fig. 7. (left) Testing error of the trained 1D model with respect to the number of snapshots using the one- and two-scale training strategies using data generated with the screened-Coulomb potential and parameters $\mu_1 = 5.0$, $\mu_2 = 0.5$ (right) normalized wall-time for the LRC and the direct all-to-all computation.

5.1. One-dimensional examples

In the first set of experiments, the domain $\Omega = [0, 5]$, $N = 20$ and $N_{\mathtt{sample}} = 1000$, where $N_{\mathtt{sample}}$ is the number of snapshots and N is the total number of points in each snapshot. For the kernel, we set α_2 and vary μ_1 to generate datasets at different interaction lengths. For each dataset we train both short-range and full-range networks using the one-scale data. The results are summarized in Table 1, where we can observe that as the characteristic interaction length increases, the accuracy of the short-range network decreases while using the full-range network can restore the accuracy. This experiment shows that local networks are often highly accurate when the interactions are localized, but the accuracy quickly deteriorates as the interaction length increases (i.e. as μ_1 decreases).

For the second set of experiments we used two sets of kernel parameters: one heavily biased towards a localized interaction length, and another in which both interaction lengths are equally weighted. For each set of kernel parameters, we generate 10, 000 small-scale snapshots using $\Omega_{\mathtt{sr}} = [0, 5]$ and $N = 20$, and a large number of large-scale snapshots using $\Omega_{\mathtt{lr}} = [0, 50]$ and $N = 200$ particles. The interaction radius $R = 1.5$, $\delta_{\mathtt{min}} = 0.05$, and $N_{\mathtt{FFT}}$ is 501. We train the network with the one- and two-scale training strategies described in the prequel. Fig. 7 (left) depicts the advantage of using the two-scale training strategy: we obtain roughly the same accuracy at a fraction of the number of large-scale training samples. We observe that when the number of large-scale training samples is sufficiently large, the resulting test accuracy is independent of the training strategy. We also observe that the training dynamic is stable with respect to different random seeds.

For the third set of experiments we measure the impact of $N_{\mathtt{FFT}}$ on the approximation error. We test a screened-Coulomb type potential with parameters $\mu_1 = 5.0$, $\mu_2 = 0.5$, $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, and $N_{\mathtt{sample}} = 1000$. The domain Ω is $[0, 50]$ and $N = 200$. We run the one-scale training procedure with varying $N_{\mathtt{FFT}}$ (the number of Fourier multipliers), starting from $N_{\mathtt{FFT}} = 63$ and doubling them until $N_{\mathtt{FFT}} = 501$. Table 2 shows that the errors are relatively insensitive to the value of $N_{\mathtt{FFT}}$. The accuracy achieved by the architecture without the LRC-layer (denoted as None in Table 2) is added in order to demonstrate that the architecture is indeed capturing the LRIs.

We compare the LRC-layer with a direct all-to-all computation. We benchmark the wall time of both layers, with increasingly number of particles. To account for implementation effects we normalize the wall times in Fig. 7 (right) and the results corroborate the complexity claims made in Section 2.

Table 3
Relative testing error for trained screened-Coulomb type 2D models with $\alpha_1 = 1, \alpha_2 = 0$, and varying μ_1. Again μ_2 can be arbitrary given that $\alpha_2 = 0$.

μ_1	1.0	2.0	5.0	10.0
short-range network	0.07847	0.02332	0.00433	0.00242
full-range network	**0.00785**	**0.00526**	**0.00363**	**0.00181**

Table 4
Error with respect to $N_{\texttt{FFT}}$ in the 2D case.

$N_{\texttt{FFT}}$	None	21^2	31^2	45^2	63^2
Relative testing error	0.01872	0.00202	0.00168	0.00153	0.00177

Table 5
Values of parameters λ_1 and λ_2 after training with respect to $N_{\texttt{FFT}}$.

	1D case				2D case			
$N_{\texttt{FFT}}$	63	125	251	501	21^2	31^2	45^2	63^2
λ_1	2.697	3.689	4.181	4.599	3.608	3.664	3.096	2.955
λ_2	0.522	0.525	0.517	0.519	0.926	1.039	1.088	1.082

5.2. Two-dimensional examples

We perform the same experiments as in the one-dimensional case. We fix $\Omega = [0, 15]^2$, $N = 450$ and $N_{\texttt{sample}} = 10000$. The results are summarized in Table 3, which shows that as μ decreases, the full-range network outperforms the short-range one.

For the second set of experiments, $R = 1.5$, $\delta_{\texttt{min}} = 0.05$, and $N_{\texttt{FFT}}$ is 31^2. For the small-scale data, $\Omega_{\texttt{sr}} = [0, 3]^2$, $N = 18$, and $N_{\texttt{sample}} = 10,000$. For the large-scale data, $\Omega_{\texttt{lr}} = [0, 15]^2$, $N = 450$. Similarly to the 1D case, we train the networks with both strategies using different amounts of large-scale data. The results summarized in Fig. 8 show that the two-scale strategy efficiently captures the LRIs with only a small number of the long-range training samples. Analogously to the 1D case, we can observe that for sufficiently large-scale training samples the resulting test accuracy is identical regardless of the training strategy used. Also similar to Fig. 7 (left), we find that the lowest achievable test error is larger in Fig. 8 (right, with a larger α_2) than that in Fig. 8 (left, with a smaller α_2). Nonetheless, we observe that the test error of the two-scale training strategy becomes less sensitive with respect to the number of training samples when α_2 becomes larger, i.e. the LRI becomes more prominent.

For the third set of tests, we use a screened-Coulomb type potential is tested with $\mu_1 = 10.0$, $\mu_2 = 1.0$, $\alpha_1 = 0.9$, $\alpha_2 = 0.1$. Here $\Omega = [0, 5]^2$, $N = 50$ and $N_{\texttt{sample}} = 1000$. Starting with $N_{\texttt{FFT}} = 21^2$, we steadily increase its value and repeat the same training procedure. The results are summarized in Table 4 where one observes the same trend as in the one-dimensional case.

We recall that the Fourier multipliers are parametrized following

$$\hat{\phi}_{\beta,\lambda}(k) = \frac{4\pi\beta}{\|k\|^2 + \lambda^2}, \tag{24}$$

where β and λ are two trainable parameters with λ providing a measure of the decay in space. Therefore, $N_{\texttt{FFT}}$ only determines the number of Fourier modes and not the parameters of the ansatz. As long as the Fourier kernel is properly sampled, the method is able to compute the correct characteristic interaction length.

One can observe this phenomenon in the experiment above, in which we extract the terminal value after training of the parameters λ_1 and λ_2 that correspond to the two channels in the LRC-layer, as summarized in Table 5. We observe that the value of λ_2 is very close to that of μ_2 (that is equal to 0.5 for 1D and 2.0 for 2D), which is responsible for the LRIs even for small values of $N_{\texttt{FFT}}$. This proves that we can accurately capture the LRIs, in particular, the learned kernel is able to match the representation of ψ^μ in Fourier domain, which is given in equation (8).

5.3. Three-dimensional examples

The domain Ω is $[0, 3]^3$ with 2 points in each of the 27 unit cells. The other parameters are the interaction radius $R = 1.0$, $\delta_{\texttt{min}} = 0.1$, and $N_{\texttt{sample}} = 1000$. The Fourier domain used is of size $N_{\texttt{FFT}} = 25^3$. The results in Table 6 demonstrate that full-range network is capable of maintaining good accuracy for a wide range of characteristic interactions lengths.

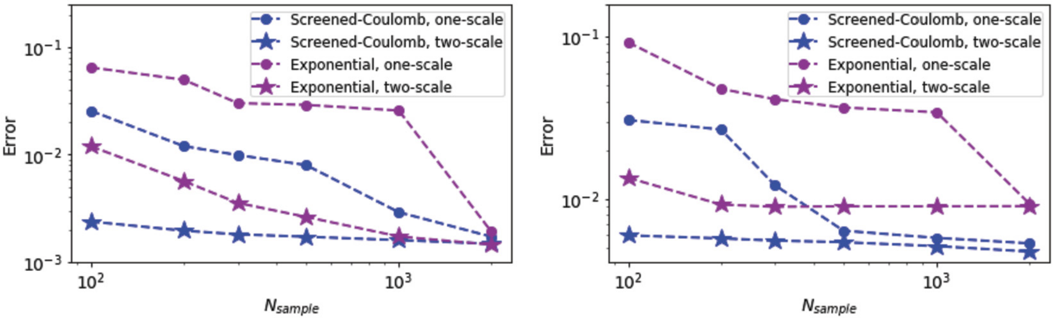


Fig. 8. Testing error of the trained 2D model with respect to the number of snapshots using the one- and two-scale training strategies using both screened-Coulomb and exponential potentials with $\mu_1 = 10$, $\mu_2 = 1$: (left) $\alpha_1 = 0.9$, and $\alpha_2 = 0.1$; and (right) $\alpha_1 = 0.5$, and $\alpha_2 = 0.5$.

Table 6
Relative testing error for trained exponential type 3D models with $\alpha_1 = 1, \alpha_2 = 0$, and varying μ_1. Again μ_2 can be arbitrary given that $\alpha_2 = 0$.

μ_1	5	7.5	10
short-range network	0.06249	0.01125	0.00175
full-range network	**0.00971**	**0.00411**	**0.00151**

6. Conclusion

We have presented an efficient long-range convolutional (LRC) layer, which leverages the non-uniform fast Fourier transform (NUFFT) to reduce the cost from quadratic to nearly-linear with respect to the number of degrees of freedom. We have also introduced a two-scale training strategy to effectively reduce the number of large-scale samples. This can be particularly important when the generation of these large-scale samples dominates the computational cost. While this paper demonstrates the effectiveness of the LRC-layer for computing the energy and force associated with a model N-body potential, we expect the LRC-layer to become a useful component in designing neural networks for modeling real chemical and materials systems, where the LRI cannot be accurately captured using short ranged models.

CRediT authorship contribution statement

Yifan Peng: Investigation, Methodology, Software, Writing, Reviewing and Editing. **Lin Lin:** Conceptualization, Investigation, Methodology, Writing, Reviewing and Editing **Lexing Ying:** Conceptualization, Investigation, Methodology, Writing, Reviewing and Editing. **Leonardo Zepeda-Nunez:** Conceptualization, Investigation, Methodology, Software, Writing, Reviewing and Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

The work of L.L. is partially supported by the Department of Energy under Grant No. DE-SC0017867 and the CAMERA program, and by the National Science Foundation under Grant No. DMS-1652330. The work of L.Y. is partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program and also by the National Science Foundation under award DMS-1818449. The work of L.Z.-N. is partially supported by the National Science Foundation under the grant DMS-2012292, and by NSF TRIPODS award 1740707.

Appendix A. Notation

A table of notations is summarized in Table A.7.

Table A.7
Symbols introduced in the current paper with their corresponding meaning.

Notation	
Symbol	Meaning
Data	
d	Spatial dimension of the problem
$\Omega = [0, L]^d \subset \mathbb{R}^d$	Computational Domain
$\{x_i\}_{i=1}^N \subset \Omega$	Point cloud
N	Number of points in the point cloud
$N_{\texttt{sample}}$	Number of snapshots for training
ψ^μ	Interaction kernel
μ	Inverse characteristic interaction length
U	Potential
F_j	Forces exerted over the j-th particle
Networks	
$\mathcal{D}^i$	Descriptor associated to x_i
$\mathcal{F}$	Fitting Network
θ	Generic trainable parameters
f_θ	Trainable function inside the descriptor
g_θ	Trainable function inside the fitting network
R	Interaction radius
LRC-layer	
g_τ	Mollifier of the Dirac deltas defined in equation (B.2)
τ	Broadening factor in the mollifier
`FFT`, `IFFT`	Fast Fourier transform and its inverse
ϕ_θ	Kernel with trainable parameters θ
$\hat{\phi}_\theta$	Fourier transform of the kernel
$L_{\texttt{FFT}}$	Number of Fourier modes per dimension
$N_{\texttt{FFT}} = L_{\texttt{FFT}}^d$.	Total number of Fourier modes

Appendix B. NUFFT

In this section we provide further details for the NUFFT implementation. Suppose that the input of the NUFFT is given by $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$, where each point has a given associated weight f_i. The first step is to construct the weighted train of Dirac deltas as

$$f(x) = \sum_{j=1}^{N} f_j \delta\left(x - x_j\right). \tag{B.1}$$

We point out that in some of the experiments f_j simply equals to 1. One then defines a periodic Gaussian convolution kernel

$$g_\tau(x) = \sum_{\ell \in \mathbb{Z}^d} e^{-\|x - \ell L\|^2/4\tau}, \tag{B.2}$$

where L is the length of the interval and τ determines the size of mollification. In practice a good choice is $\tau = 12(\frac{L}{2\pi L_{\texttt{FFT}}})^2$ [14], where $L_{\texttt{FFT}}$ is the number of points in each dimension and $N_{\texttt{FFT}} = L_{\texttt{FFT}}^d$. We define

$$f_\tau(x) = f * g_\tau(x) = \int_{[0,L]^d} f(y) g_\tau(x - y) dy = \sum_{j=1}^{N} f_j g_\tau(x - x_j). \tag{B.3}$$

With the Fourier transform defined as

$$F_\tau(k) = \frac{1}{L^d} \int_{[0,L]^d} f_\tau(x) e^{-i2\pi k \cdot x/L} dx \tag{B.4}$$

for $k \in \mathbb{Z}^d$, we compute its discrete counterpart

$$F_\tau(k) \approx \frac{1}{N_{\texttt{FFT}}} \sum_{m \in [0, L_{\texttt{FFT}}-1]^d} f_\tau\left(Lm/L_{\texttt{FFT}}\right) e^{-i2\pi k \cdot m/L_{\texttt{FFT}}} \tag{B.5}$$

$$\approx \frac{1}{N_{\mathrm{FFT}}} \sum_{m\in[0,L_{\mathrm{FFT}}-1]^d} \sum_{j=1}^{N} f_j g_\tau \left(Lm/L_{\mathrm{FFT}} - x_j\right) e^{-i2\pi k\cdot m/L_{\mathrm{FFT}}} \tag{B.6}$$

This operation can be done in $\mathcal{O}(N_{\mathrm{FFT}} \log(N_{\mathrm{FFT}}))$ steps, independently of the number of inputs. Once this is computed, one can compute the Fourier transform of f at each frequency point by

$$F(k) = \left(\frac{\pi}{\tau}\right)^{d/2} e^{\|k\|^2\tau} F_\tau(k) \tag{B.7}$$

Once the Fourier transform of the Dirac delta train is ready, we multiply it by the Fourier multiplier $\hat{\phi}(k)$, which is the Fourier transform of ϕ:

$$\hat{v}(k) = \hat{\phi}(k) F(k) \tag{B.8}$$

In the next sage, one needs to compute the inverse transform, and evaluate into the target points $\{x_i\}$. First we deconvolve the signal

$$\hat{v}_{-\tau}(k) = \left(\frac{\pi}{\tau}\right)^{d/2} e^{\|k\|^2\tau} \hat{v}(k) \tag{B.9}$$

and compute the inverse Fourier transform

$$u_{-\tau}(x) = \sum_{k\in[0,N_{\mathrm{FFT}}-1]^d} \hat{v}_{-\tau}(k) e^{ik\cdot x}. \tag{B.10}$$

Next, we interpolate to the point cloud

$$u\left(x_j\right) = u_{-\tau} * g_\tau\left(x_j\right) = \frac{1}{L^d} \int\limits_{[0,L]^d} u_{-\tau}(x) g_\tau\left(x_j - x\right) dx \tag{B.11}$$

$$\approx \frac{1}{N_{\mathrm{FFT}}} \sum_{m\in[0,L_{\mathrm{FFT}}-1]^d} u_{-\tau}\left(Lm/L_{\mathrm{FFT}}\right) g_\tau\left(x_j - Lm/L_{\mathrm{FFT}}\right) \tag{B.12}$$

Even though in the current implementation all the parameters of the NUFFT are fixed, they can in principle be trained along with the rest of the networks. This training, if done naively increases significantly the computational cost. How to perform this operation efficiently is a direction of future research.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] M. Aubry, U. Schlickewei, D. Cremers, The wave kernel signature: a quantum mechanical approach to shape analysis, in: 2011 IEEE International Conference on Computer Vision Workshops (ICCV), 2011, pp. 1626–1633.

[3] A.H. Barnett, J. Magland, L. af Klinteberg, A parallel nonuniform fast Fourier transform library based on an "exponential of semicircle" kernel, SIAM J. Sci. Comput. 41 (5) (2019) C479–C504.

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

[5] T. Bereau, R.A. DiStasio, A. Tkatchenko, O.A. von Lilienfeld, Non-covalent interactions across organic and biological subsets of chemical space: physics-based potentials parametrized from machine learning, J. Chem. Phys. 148 (24) (2018) 241706.

[6] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv:1312.6203.

[7] D. Chen, X. Tian, Y. Shen, O. Ming, On visual similarity based 3D model retrieval, Comput. Graph. Forum 22 (3) (2003) 223–232.

[8] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3D object detection network for autonomous driving, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6526–6534.

[9] Taco Cohen, Max Welling, Group equivariant convolutional networks, in: Maria Florina Balcan, Kilian Q. Weinberger (Eds.), Proceedings of the 33rd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 48, New York, New York, USA, 20–22 Jun 2016, PMLR, 2016, pp. 2990–2999.

[10] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (90) (1965) 297–301.

[11] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems 29, 2016, pp. 3844–3852.

[12] H. Deng, T. Birdal, S. Ilic, PPF-FoldNet: unsupervised learning of rotation invariant 3D local descriptors, in: Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[13] Z. Deng, C. Chen, X.G. Li, S.P. Ong, An electrostatic spectral neighbor analysis potential for lithium nitride, NPJ Comput. Mater. 5 (2019).

[14] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (6) (1993) 1368–1393.

[15] Y. Fan, J. Feliu-Fabà, L. Lin, L. Ying, L. Zepeda-Núñez, A multiscale neural network based on hierarchical nested bases, Res. Math. Sci. 6 (2) (Mar 2019) 21.

[16] Y. Fan, L. Lin, L. Ying, L. Zepeda-Núñez, A multiscale neural network based on hierarchical matrices, Multiscale Model. Simul. 17 (4) (2019) 1189–1213.

[17] Y. Fan, L. Ying, Solving optical tomography with deep learning, arXiv:1910.04756, 2019.

[18] Roger H. French, V. Adrian Parsegian, Rudolf Podgornik, Rick F. Rajter, Anand Jagota, Jian Luo, Dilip Asthagiri, Manoj K. Chaudhury, Yet Ming Chiang, Steve Granick, Sergei Kalinin, Mehran Kardar, Roland Kjellander, David C. Langreth, Jennifer Lewis, Steve Lustig, David Wesolowski, John S. Wettlaufer Wai Yim Ching, Mike Finnis, Frank Houlihan, O. Anatole Von Lilienfeld, Carel Jan Van Oss, Thomas Zemb, Long range interactions in nanoscale science, Rev. Mod. Phys. 82 (2) (2010) 1887–1944.

[19] M. Gadelha, R. Wang, S. Maji, Multiresolution tree networks for 3d point cloud processing, in: Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[20] L. Greengard, J. Lee, Accelerating the nonuniform fast Fourier transform, SIAM Rev. 46 (3) (2004) 443–454.

[21] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325–348.

[22] A. Grisafi, M. Ceriotti, Incorporating long-range physics in atomic-scale machine learning, J. Chem. Phys. 151 (20) (2019) 204105.

[23] A. Grisafi, J. Nigam, M. Ceriotti, Multi-scale approach for the prediction of atomic scale properties, arXiv:2008.12122.

[24] M. Hirn, S. Mallat, N. Poilvert, Wavelet scattering regression of quantum chemical energies, Multiscale Model. Simul. 15 (2) (2017) 827–863.

[25] H. Huo, M. Rupp, Unified representation of molecules and crystals for machine learning, arXiv:1704.06439.

[26] V. Jampani, M. Kiefel, P.V. Gehler, Learning sparse high dimensional filters: image filtering, dense CRFs and bilateral neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[27] John Edward Jones, On the determination of molecular fields, in: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 106, The Royal Society, 1924, pp. 463–477.

[28] Y. Khoo, L. Ying, SwitchNet: a neural network model for forward and inverse scattering problems, SIAM J. Sci. Comput. 41 (5) (2019) A3182–A3201.

[29] D. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations (ICLR), May 2015.

[30] R. Klokov, V. Lempitsky, Escape from cells: deep Kd-networks for the recognition of 3D point cloud models, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 863–872.

[31] T.W. Ko, J.A. Finkler, S. Goedecker, J. Behler, A fourth-generation high-dimensional neural network potential with accurate electrostatics including non-local charge transfer, arXiv:2009.06484, 2009.

[32] R. Kondor, N. Teneva, V. Garg, Multiresolution matrix factorization, in: Proceedings of Machine Learning Research, vol. 32, 2014, pp. 1620–1628.

[33] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, PointCNN: convolution on x-transformed points, Adv. Neural Inf. Process. Syst. 31 (2018) 820–830.

[34] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations, arXiv:2006.09535, 2020.

[35] Y. Liu, B. Fan, S. Xiang, C. Pan, Relation-shape convolutional neural network for point cloud analysis, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 8887–8896.

[36] D. Maturana, S. Scherer, Voxnet: a 3D convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922–928.

[37] J. Nigam, S. Pozdnyakov, M. Ceriotti, Recursive evaluation and iterative contraction of N-body equivariant features, J. Chem. Phys. 153 (12) (2020) 121101.

[38] Y.J. Oh, Y. Watanabe, Development of small robot for home floor cleaning, in: Proceedings of the 41st SICE Annual Conference, vol. 5, 2002, pp. 3222–3223.

[39] Y. Park, V. Lepetit, W. Woo, Multiple 3D object tracking for augmented reality, in: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008, pp. 117–120.

[40] David Pfau, James S. Spencer, Alexander G.D.G. Matthews, W.M.C. Foulkes, Ab initio solution of the many-electron Schrödinger equation with deep neural networks, Phys. Rev. Res. 2 (Sep 2020) 033429.

[41] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85.

[42] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: deep hierarchical feature learning on point sets in a metric space, Adv. Neural Inf. Process. Syst. 30 (2017) 5099–5108.

[43] G. Riegler, A.O. Ulusoy, A. Geiger, Octnet: learning deep 3D representations at high resolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6620–6629.

[44] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 234–241.

[45] K. Rossi, V. Jurásková, R. Wischert, L. Garel, C. Corminboeuf, M. Ceriotti, Simulating solvation and acidity in complex mixtures with first-principles accuracy: the case of CH_3SO_3H and H_2O_2 in phenol, J. Chem. Theory Comput. 16 (8) (2020) 5139–5149.

[46] M. Rupp, A. Tkatchenko, K. Müller, O.A. Von Lilienfeld, Fast and accurate modeling of molecular atomization energies with machine learning, Phys. Rev. Lett. 108 (5) (2012) 058301.

[47] R.B. Rusu, N. Blodow, Z.C. Marton, M. Beetz, Aligning point cloud views using persistent feature histograms, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 3384–3391.

[48] R.B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (FPFH) for 3D registration, in: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 1848–1853.

[49] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, A. Tatsuma, S. Thermos, A. Axenopoulos, G.Th. Papadopoulos, P. Daras, X. Deng, Z. Lian, B. Li, H. Johan, Y. Lu, S. Mk, Large-scale 3D shape retrieval from shapenet core55, in: Eurographics Workshop on 3D Object Retrieval, 2016.

[50] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, J. Kautz, SPLATNet: Sparse lattice networks for point cloud processing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2530–2539.

[51] L. Wang, Y. Huang, Y. Hou, S. Zhang, J. Shan, Graph attention convolution for point cloud semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[52] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D shapenets: a deep representation for volumetric shapes, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1912–1920.

[53] M. Xu, W. Dai, Y. Shen, H. Xiong, MSGCNN: multi-scale graph convolutional neural network for point cloud segmentation, in: 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), 2019, pp. 118–127.

[54] Y. Xu, T. Fan, M. Xu, L. Zeng, Y. Qiao, SpideCNN: deep learning on point sets with parameterized convolutional filters, in: Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[55] Y. Yang, C. Feng, Y. Shen, D. Tian, FoldingNet: point cloud auto-encoder via deep grid deformation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 206–215.

[56] K. Yao, J.E. Herr, D.W. Toth, R. Mckintyre, J. Parkhill, The tensorMol-0.1 model chemistry: a neural network augmented with long-range physics, Chem. Sci. 9 (2018) 2261–2269.

[57] X. Ye, J. Li, H. Huang, L. Du, X. Zhang, 3D recurrent neural networks with context fusion for point cloud semantic segmentation, in: Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[58] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R.R. Salakhutdinov, A.J. Smola, Deep sets, Adv. Neural Inf. Process. Syst. (2017) 3391–3401.
[59] W. Zeng, T. Gevers, 3DContextNet: K-d tree guided hierarchical learning of point clouds using local and global contextual cues, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, September 2018.
[60] L. Zepeda-Núñez, Y. Chen, J. Zhang, W. Jia, L. Zhang, L. Lin, Deep density: circumventing the Kohn-Sham equations via symmetry preserving neural networks, https://www.math.wisc.edu/~lzepeda/Deep-Density.pdf, 2019.
[61] Z. Zhai, X. Zhang, L. Yao, Multi-scale dynamic graph convolution network for point clouds classification, IEEE Access 8 (2020) 65591–65598.
[62] Jiefu Zhang, Leonardo Zepeda-Núñez, Yuan Yao, Lin Lin, Learning the mapping $\mathbf{x} \mapsto \sum_{i=1}^{d} x_i^2$: the cost of finding the needle in a haystack, Commun. Appl. Math. Comput. Sci. (2020).
[63] L. Zhang, M. Chen, X. Wu, H. Wang, W. E, R. Car, Deep neural network for the dielectric response of insulators, arXiv:1906.11434, 2019.
[64] L. Zhang, J. Han, H. Wang, R. Car, W. E., Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics, Phys. Rev. Lett. 120 (Apr 2018) 143001.
[65] L. Zhang, J. Han, Ha. Wang, W. Saidi, R. Car, W. E., End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems, Adv. Neural Inf. Process. Syst. 31 (2018) 4441–4451.
[66] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, Weinan E., Deepcg: Constructing coarse-grained models via deep neural networks, J. Chem. Phys. 149 (3) (2018) 034101.
[67] Linfeng Zhang, Han Wang, Maria Carolina Muniz, Athanassios Z. Panagiotopoulos, Roberto Car, E. Weinan, A deep potential model with long-range electrostatic interactions, J. Chem. Phys. 156 (12) (2022) 124107.
[68] Y. Zhou, O. Tuzel, Voxelnet: End-to-end learning for point cloud based 3D object detection, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4490–4499.