Solving Newton's equations of motion with large timesteps using recurrent neural networks based operators

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 025002 https://doi.org/10.1088/2632-2153/ac5f60

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
13 December 2021

REVISED
10 March 2022

ACCEPTED FOR PUBLICATION
21 March 2022

PUBLISHED
5 April 2022

PAPER

Solving Newton's equations of motion with large timesteps using recurrent neural networks based operators

J C S Kadupitiya, Geoffrey C Fox and Vikram Jadhao*

Intelligent Systems Engineering, Indiana University, Bloomington, IN 47408, United States of America
* Author to whom any correspondence should be addressed.

E-mail: vjadhao@iu.edu

Keywords: molecular dynamics, deep learning, recurrent neural networks, Newton's equations, time evolution operators, machine learning

Abstract

Classical molecular dynamics simulations are based on solving Newton's equations of motion. Using a small timestep, numerical integrators such as Verlet generate trajectories of particles as solutions to Newton's equations. We introduce operators derived using recurrent neural networks that accurately solve Newton's equations utilizing sequences of past trajectory data, and produce energy-conserving dynamics of particles using timesteps up to 4000 times larger compared to the Verlet timestep. We demonstrate significant speedup in many example problems including 3D systems of up to 16 particles.

1. Introduction

Newton's equations of motion [1] are the basis of powerful computational methods such as classical molecular dynamics (MD) that are used to understand the microscopic origins of a wide range of material and biological phenomena [2, 3]. In the MD method, Newton's equations are integrated for a system of many particles using numerical integrators such as Verlet [4] to produce particle trajectories. The time evolution is performed one small timestep at a time for long times to accurately sample enough representative configurations in order to extract useful information. Consider the 2nd order ordinary Verlet integrator $\vec{x}(t+\Delta) = 2\vec{x}(t) - \vec{x}(t-\Delta) + \Delta^2 \vec{f}(t)/m$ that updates the current position $\vec{x}(t)$ of a particle of mass m at time t to position $\vec{x}(t+\Delta)$ after timestep Δ using the previous position $\vec{x}(t-\Delta)$ and the force $\vec{f}(t)$ at time t. This integrator produces an error of $O(\Delta^4)$ in each local update and incurs a global error of $O(\Delta^2)$ [3, 5]. These errors are reduced by choosing a small Δ which often makes the simulations computationally expensive.

The ordinary Verlet integrator requires a sequence of two positions $(\vec{x}_{t-\Delta}, \vec{x}_t)$ to update the particle position using other quantities (e.g. $\vec{f}$ and m). These quantities can be inferred using the information encoded in a long sequence of positions such that the time evolution can be done with only the history of positions as input. We illustrate this with a 1-dimensional example of a particle experiencing simple harmonic motion governed by the force $f = -kx$. One can show that the particle position can be evolved to $t+\Delta$ using a sequence of three positions via the function $\mathcal{V} = x_{t-\Delta}^{-1}\left(x_t^2 - x_{t-\Delta}^2 + x_t x_{t-2\Delta}\right)$, which also incurs a global error of $O(\Delta^2)$. This idea generalizes for higher-order integrators [6] and many-particle systems such that the time evolution can be performed via $\mathcal{V}(\vec{x}_t, \vec{x}_{t-\Delta}, \ldots, \vec{x}_{t-s\Delta})$ that takes a sequence of s positions. The longer history of input positions enables integrators to perform accurate time evolution with a larger Δ, however, generally at the expense of higher computing costs per timestep.

The use of deep learning in sequence processing and time series prediction problems has been well studied by the industry for different applications including voice recognition and translation [7], pattern recognition in stock market data [8], and ride-hailing [9]. Recurrent neural networks (RNNs) are established deep learning tools in these applications. In this work, we develop RNN based operators to perform accurate time evolution of one-particle and few-particle systems utilizing sequences of past trajectories of particles. The RNN-based operators are trained using the ground truth results obtained with the Verlet integrator.

They possess a complex mathematical structure described with up to 100 000 parameters. We demonstrate that the network complexity enables the operators to perform time evolution of systems of up to 16 particles for a wide range of force fields using timesteps that are up to 4000× larger than the baseline Verlet timestep. The relatively small time for inferring the positions as predictions of the deep learning model keeps overhead costs low and we demonstrate significant net speedups for larger timesteps.

Recent years have seen a surge in the use of machine learning to enhance the performance and usability of MD simulations, for example, see reviews [10–12]. Machine learning has been used to accelerate the sampling of rugged free-energy landscapes [13], generate new configuration updates in reduced-dimensional space [14], learn MD force fields [15], auto-tune simulation timestep [16], classify particle assembly landscapes [17], characterize material structures [18, 19], and derive 'surrogates' for MD simulations [20–24]. Of particular relevance to our work are the deep learning approaches that learn differential equations and replicate the outputs of numerical integrators used in MD simulations [25, 26, 26–35]. Most of these approaches have focused on learning ordinary differential equations describing the dynamics of simple 1D systems and replicating the outputs of associated numerical integrators using baseline timestep values [25–30, 35]. Some studies have probed the use of deep learning methods to learn partial differential equations [25, 26, 32, 33] and, more recently, solve them using discretization steps larger than the baseline timestep [34, 35]. For example, Shen *et al* [34] used an artificial neural network based model as a corrector to reduce the local truncation error associated with the Euler integrator, and demonstrated time evolution of simple two-body problems with timesteps up to 200 × the baseline Euler timestep. A 100-fold increase in the timestep led to over three orders of magnitude increase in the error. In this work, we develop a deep learning approach that utilizes RNNs to replicate the output of Verlet integrators for a variety of force fields and perform accurate time evolution of systems of up to 16 particles in 3D using timesteps that are up to 4000× larger than the baseline Verlet timestep. We obtain state-of-the-art results in terms of the timesteps, the number of particles, and the complexity of the potential characterizing the interactions between particles.

We note that RNNs have been recently used to determine the evolution of configurations described by a few collective variables characterizing the system dynamics [36]. Other related work has focused on the goal of producing energy-conserving Hamiltonians by training neural networks in an unsupervised manner to observe the positions and momenta of many-particle systems in 1D [30, 37–40]. On the other hand, we note the active work in the development of approaches that do not rely on deep learning, such as the multiple timestep methods, for simulating the dynamics of complex systems with large timesteps [41–44].

2. Recurrent neural network based operators for predicting dynamics of particles

Figure 1 shows the overview of our deep learning approach to learn the time evolution operator that evolves the dynamics of an N-particle system. We begin by selecting the potential energy function governing the dynamics of the particles. In addition to the potential energy, the N-particle system is specified by particle masses and the initial positions and velocities of the particles. The attributes of the N-particle system together with the Verlet timestep Δ form the input. The input system attributes are fed to the Verlet integrator to simulate the dynamics with timestep Δ up to $S_{\rm V}$ computational steps. Out of the full trajectory data (e.g. positions and velocities) up to $S_{\rm V}$ steps, $S_{\rm R}$ number of configurations (frames) separated by $\Delta_{\rm R}$ are distilled. Note that this requirement to kickstart the time evolution using the Verlet integrator enforces $S_{\rm V} = \Delta_{\rm R}(S_{\rm R} - 1)/\Delta$. Using this initial sequence of particle configurations of length $S_{\rm R}$, a trained RNN based operator $\mathscr{R}$ predicts the time evolution of the system after timestep $\Delta_{\rm R}$. Then, the input sequence to $\mathscr{R}$ is left-shifted to discard the oldest time frame, and the latest frame predicted by $\mathscr{R}$ is appended to the right-end of the sequence. The adjusted input sequence is fed back to $\mathscr{R}$ to evolve the system $\Delta_{\rm R}$ further in time and the same process is repeated until the end of the simulation. The time evolution results in the output comprising the trajectories of the particles.

As shown in figure 1, $\mathscr{R}$ is trained using the ground-truth particle trajectories generated via the velocity Verlet integrator with small timestep $\Delta = 0.001$ for the system specified by the selected potential energy function. The velocity Verlet integrator updates the configuration of particles via two steps, which we describe for the case of a single particle; extension to many particles is straightforward. First, the position $\vec{x}(t)$ of a particle of mass m at time t is evolved a timestep Δ forward in time:

$$\vec{x}(t+\Delta) = \vec{x}(t) + \Delta\vec{v}(t) + 0.5\Delta^2\,\vec{f}(t)/m, \tag{1}$$

where $\vec{v}(t)$ and $\vec{f}(t)$ are the current velocity and force at time t respectively. Next, the velocity $\vec{v}(t)$ at time t is updated to $\vec{v}(t+\Delta)$:

$$\vec{v}(t+\Delta) = \vec{v}(t) + 0.5\,(\Delta/m)\left(\vec{f}(t) + \vec{f}(t+\Delta)\right), \tag{2}$$

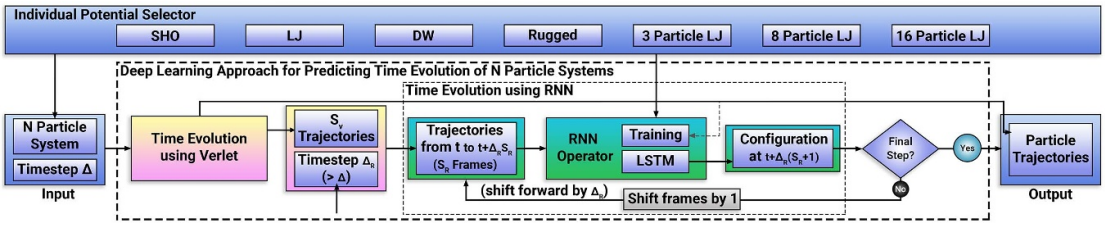

Figure 1. Overview of the deep learning approach involving recurrent neural networks (RNN) based operators to solve Newton's equations of motion and predict dynamics of N particles.

where $\vec{f}(t+\Delta)$ is the force computed at time $t+\Delta$ using the updated position of the particle evaluated in equation (1). The time evolution moves forward following equations (1) and (2) with $\vec{x}(t+\Delta)$ and $\vec{v}(t+\Delta)$ as current position and velocity respectively.

We emphasize that our approach trains separate RNNs for furnishing the time evolution of systems described by different functional forms of potential energy. For example, if a 1D simple harmonic potential $(1/2)kx^2$ is selected as the potential energy function, $\mathscr{R}$ learns to predict the dynamics of one particle in a harmonic potential for unseen values of k, however, it does not learn to predict the time evolution of a particle in a qualitatively different potential energy such as a double well potential. The RNN based operators are at the heart of our approach. In order to understand how these operators are designed and trained, we first briefly describe the key characteristics of RNNs.

2.1. Recurrent neural networks

Recurrent neural networks (RNNs) process input sequence data and maintain a vector $\vec{h}_t$ known as the 'hidden state' for each recurrent cell to model the temporal behavior of sequences through directed cyclic connections between cells. $\vec{h}_t$ is updated by applying a function F to the previous hidden state $(\vec{h}_{t-1})$ and the current input $(\vec{x}_t)$. The cells are arranged in a fashion where they fire when the right sequence is fed. A common choice for F is the Long Short Term Memory (LSTM) units [45]. There are several architectures of LSTM units. An often employed architecture consists of a cell (the memory part of the LSTM unit) and three 'regulators', usually called gates, that regulate the flow of information inside the LSTM unit. An input gate (i_t) controls how much new information is added from the present input (x_t) and past hidden state (h_{t-1}) to the present cell state (c_t). A forget gate (f_t) decides what is removed or retained and carried forward to c_t from the previous cell state (c_{t-1}). An output gate (o_t) decides what to output as the current hidden state (h_t) from the current cell state. The LSTM formulation can be expressed as:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ h_t &= o_t \circ \sigma_h(c_t). \end{aligned} \tag{3}$$

Here, $x_t \in \mathbf{R}^d$ is the input vector to the LSTM unit, $f_t \in \mathbf{R}^h$ is the forget gate's activation vector, $i_t \in \mathbf{R}^h$ is the input gate's activation vector, $o_t \in \mathbf{R}^h$ is the output gate's activation vector, $h_t \in \mathbf{R}^h$ is the hidden state vector also known as the output vector of the LSTM unit, $\tilde{c}_t \in \mathbf{R}^h$ is the cell input activation vector, $c_t \in \mathbf{R}^h$ is the cell state vector, and $\circ$ is the Hadamard product operator. $W \in \mathbf{R}^{h\times d}$ and $U \in \mathbf{R}^{h\times h}$ are the weight matrices and $b \in \mathbf{R}^h$ are the bias vector parameters which need to be learned during training. σ_g and σ_h represent sigmoid function and hyperbolic tangent functions respectively. d and h refer to the number of input features and the number of hidden units respectively.

We now describe how RNNs with LSTMs can be used to design operators that process sequences of particle configurations to evolve the associated system forward in time.

2.2. RNN-based time evolution operators

We design operators $\mathscr{R}$ using RNNs with LSTM units that process a sequence of past positions and velocities as input and generate the future positions and velocities of the particles. Each component of the particle position and velocity vectors is identified as a feature. The feature size associated with the inputs and outputs for N particles in $\mathcal{D}$ physical dimensions is $d = N \times \mathcal{D} \times 2$. For example, for $N = 16$ particles interacting in

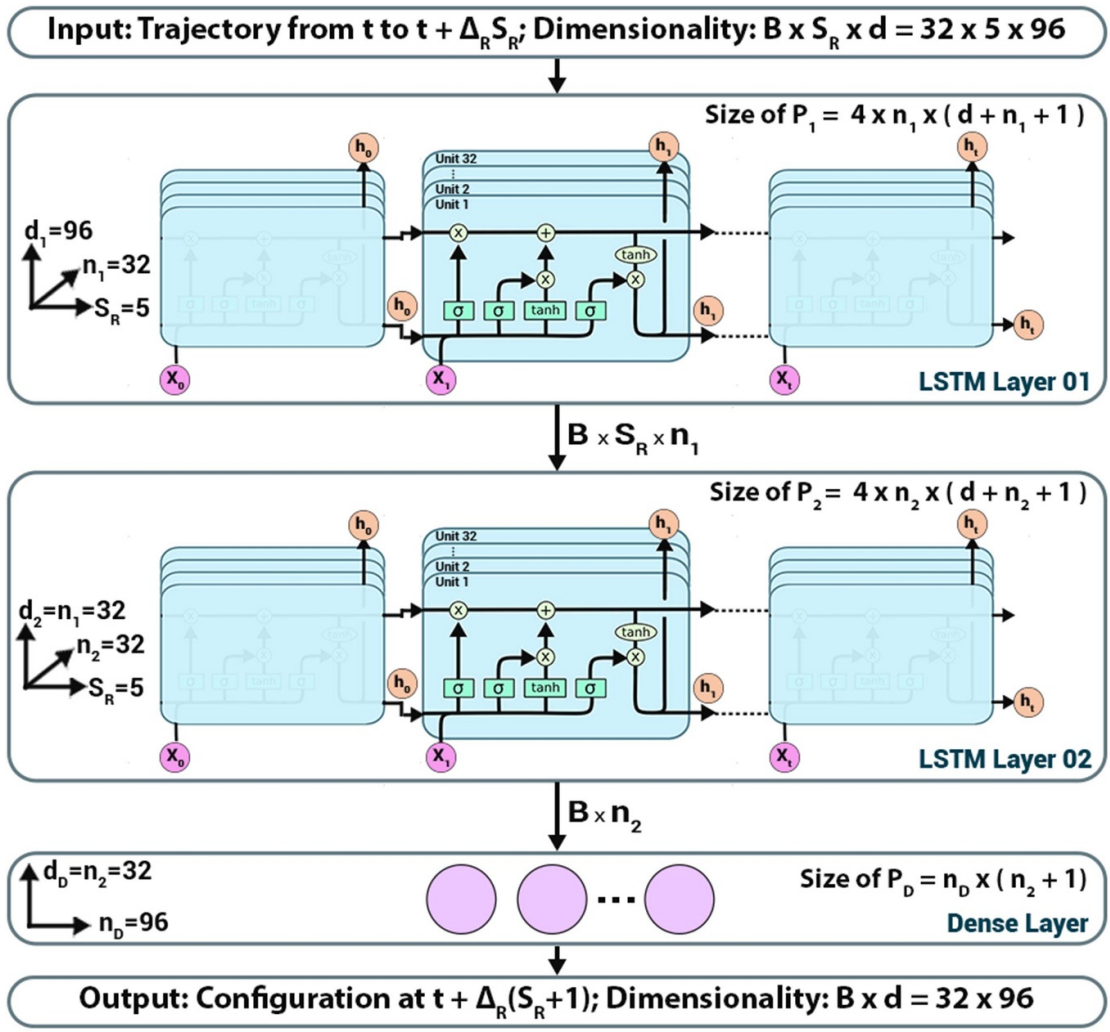

Figure 2. RNN operator performing the time evolution of an N-particle system characterized by features of size d. The operator evolves the system one Δ_R timestep forward in time using a sequence of length S_R of past trajectory states. The update is shown for an input system of $N = 16$ particles in 3D for which the feature size is $d = 96$. Parameters associated with long short term memory (LSTM) units are also shown. These parameters are defined in the main text.

3D, $d = 96$. For a system specified by the selected potential energy function, operator $\mathscr{R}$ predicts the future position and velocity vectors of the particles at time $t + \Delta_R$ by employing a sequence of length S_R of positions and velocities $\{x, v\} = \vec{x}_t, \vec{v}_t, \vec{x}_{t-\Delta_R}, \vec{v}_{t-\Delta_R}, \ldots, \vec{x}_{t-S_R\Delta_R}, \vec{v}_{t-S_R\Delta_R}$ up to time t. $\mathscr{R}$ is expressed as $\mathscr{R}[\{x, v\}] = \mathscr{D}[\mathscr{L}_2[\mathscr{L}_1[\{x, v\}]]]$, where $\mathscr{D}$, $\mathscr{L}_1$, $\mathscr{L}_2$ are the operators associated with the dense layer, the first LSTM layer, and the second LSTM layer of the RNN respectively.

The layers are sequentially connected to each other (figure 2) such that the output of one (e.g. $\mathscr{L}_1$) becomes the input for another ($\mathscr{L}_2$). Each LSTM layer consists of n number of LSTM units and contains a set of parameters in the form of weights, biases, and activation functions. For example, $\mathscr{L}_1$ has n_1 LSTM units and is characterized with weights W and U, and bias b. It takes input feature vector $\{x, v\}$ and outputs hidden state vectors $\{h\}$ which are fed as input to the $\mathscr{L}_2$ layer characterized with its own set of weights and biases. A similar connection is made between $\mathscr{L}_2$ and the dense layer $\mathscr{D}$. Post training, these layers acquire optimal values for all the parameters, and the operator $\mathscr{R}$ emerges as:

$$(\vec{x}_{t+\Delta_R}, \vec{v}_{t+\Delta_R}) = \mathscr{D}\left[\mathscr{L}_2\ \left[\mathscr{L}_1\ \left[\{x, v\}, \{P_1\}\right], \{P_2\}\right], \{P_D\}\right], \tag{4}$$

where $\{P_1\}$, $\{P_2\}$, $\{P_D\}$ are optimized parameters associated with LSTM layer 1, LSTM layer 2, and the dense layer respectively. $\mathscr{R}$ has a complex mathematical structure characterized with up to 100 000 parameters.

A similar process can be used to design operators that take a sequence of past positions as input and generate the future positions of the particles.

2.3. Operator training and implementation details

We now discuss the details of the training and implementation of RNN-based operators. These operators are created in TensorFlow with LSTM layer 1, LSTM layer 2, and final dense layer of sizes (number of hidden units) n_1, n_2, and n_D respectively. While training an operator $\mathscr{R}$ for a specific potential energy function, a $B \times S_R \times d$ dimensional vector comprising a sequence of positions and velocities $\{x, v\}$ is fed to an operator $\mathscr{R}$ as input. Here, B is a training parameter denoting the batch size, d is the feature size, and S_R is the aforementioned sequence length. During the testing phase, $B = 1$. All the parameters $\{P\}$ including the weights and biases describing the layers are optimized with an error backpropagation algorithm, implemented via stochastic gradient descent. Adam optimizer is used to optimize the error backpropagation. Outputs of the LSTM layers are wrapped with the tanh activation function. No activation function is used in the final dense layer. The mean square error (MSE) between target and predicted trajectories is used for error calculation.

The parameters $\{P\}$ of the operator $\mathscr{R}$ are saved and loaded using Keras library [46]. Values of n_1, n_2, and n_D are chosen depending on the problem complexity and data dimensions. For example, in the case of 16 particles interacting with Lennard–Jones (LJ) potential in 3D with periodic boundary conditions (feature size $d = 96$), by performing a grid search of the parameters $\{P\}$ using scikit-learn library [47], hyperparameters such as the number of units for each of the two LSTM layers (n_1, n_2), number of units in the final dense layer (n_D), batch size (B), and the number of epochs are optimized to 32, 32, 96, 256, and 2500 respectively. The learning rate of Adam optimizer is set to 0.0005 and the dropout rate is set to 0.15 to prevent overfitting. Both learning and dropout rates are selected using a trial-and-error process. The weights in the hidden layers and in the output layer are initialized for better convergence using a Xavier normal distribution [48].

Standard practices are followed to train the RNN-based operators to accurately predict trajectories while avoiding overfitting. First, the operator $\mathscr{R}$ is trained using all the training data. As expected, this model is generated in the overfitted region and it predicts results with small errors for the training samples but does not provide the same accuracy for the validation data. Next, we progressively constrain the model by reducing the number of parameters and introducing dropouts, until we obtain a similar level of low errors for samples in both training and validation datasets. Any signature of overfitting the RNN model would result in the trajectory predictions for systems in the training dataset with much lower errors compared to the errors for predictions associated with systems in the validation dataset.

We experimentally find the minimum number of hyperparameters required to keep the RNN models well-generalized and avoid overfitting by finding the optimum point in the bias-variance risk curve for the training and testing error, and introducing dropout regularization between intermediate layers of the RNN while training. Large errors obtained during the prediction of trajectories in the validation and testing datasets also alert us to the case of insufficient training samples. In general, we find that the required number of training samples depends on the complexity of the potential energy landscape. For example, in the case of 1D systems, the number of training samples required to train operator $\mathscr{R}$ to predict the trajectory of a particle in the rugged potential is 1.3 times the training samples needed by the operator $\mathscr{R}$ designed to predict the dynamics of the same particle in a double well potential. Similarly, training the RNN operator to accurately predict the dynamics of the 3D many-particle systems required 10 times more training samples compared to the operators trained to predict dynamics for 1D systems.

Prototype implementation of RNN-based operators written using Python and C++ is publicly available on GitHub [49].

3. Results and discussion

One particle experiments in 1D are performed for four potential energy functions: simple harmonic, double well, LJ, and rugged (see appendix, figure 9). Experiments on $N-$particle systems with $N = 3, 8, 16$ particles in 3D are performed on particles interacting with LJ potentials in a cubic simulation box with periodic boundary conditions. We adopt units such that the input parameters and predicted quantities are around 1.

In all experiments, RNN-based operators trained with a sequence length of $S_R = 5$ are used to perform the time evolution. Operators trained with smaller $S_R = 3$ or 4 are only able to accurately propagate the dynamics for timestep $\Delta_R \lesssim 10\Delta$, where Δ is the baseline Verlet timestep (see appendix, figure 10). Trained with $S_R = 5$, operators produce accurate dynamics for timesteps up to 4000Δ.

3.1. One particle systems in 1D

Our first set of experiments focus on training and testing the RNN-based $\mathscr{R}$ operators to predict the dynamics of one particle systems in 1D. Results are shown for dynamics predicted by four $\mathscr{R}$ operators for four representative one particle systems, each characterized with a different 1D potential. For each of these

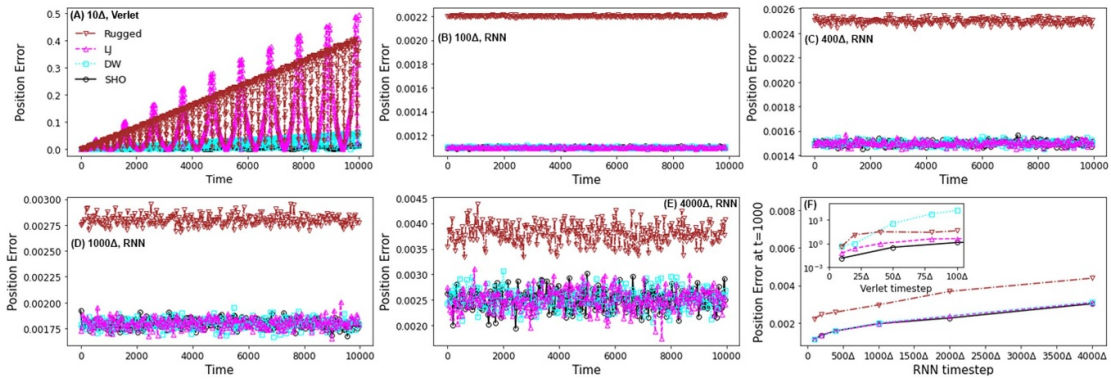

Figure 3. Errors incurred in the predictions of particle positions made by Verlet integrator and four RNN operators for four 1D systems. Black circles, blue squares, magenta up triangles, and brown down triangles represent errors in position predictions for a particle in a simple harmonic potential, double well potential, Lennard–Jones potential, and rugged potential respectively (see the main text for a detailed description of these 1D systems). (A) Errors as a function of time for dynamics extracted using the Verlet integrator with timestep $dt = 10\Delta$. (B)–(E) Errors as a function of time when the time evolution is performed by the RNN operators using timestep $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$ and 4000Δ respectively. (F) Errors in position predictions by the RNN operators at time $t = 1000$ as a function of timestep for the same four 1D systems (inset shows results on a log scale obtained using the Verlet integrator).

one particle systems in 1D, the training and validation datasets are generated by recording the dynamics associated with a few discrete initial configurations, particle masses, and, in some cases, parameters characterizing the potential energy. We describe the process in detail for the case of a particle in a simple harmonic potential $U = (1/2)kx^2$. A similar process is followed for all other 1D systems (see appendix for details).

For the 1D system of a particle in a simple harmonic potential, the dataset consists of ground-truth trajectories associated with input systems generated by sweeping over five discrete values of initial position of the particle ($x_0 = -10, -8, -6, -4, -2$), 10 discrete values of particle mass ($m = 1, 2, \ldots, 9, 10$), and 10 discrete values of spring constant ($k = 1, 2, \ldots, 9, 10$). The trajectory data for each of these input systems is obtained using simulations performed with the Verlet integrator with $\Delta = 0.001$ up to time $t = 200$. This process generates a dataset of 500 simulations, each having 400 000 position and velocity values. The entire dataset is randomly shuffled and separated into training and validation sets using a ratio of 0.8:0.2. In other words, 80% of the simulations (400 systems) are selected randomly as part of the training dataset, and the remaining 20% (100 systems) are separated into the validation dataset. The testing dataset to evaluate the predictions of the RNN-based operator comprises 100 input systems characterized with m, x_0, k values distinct from those used in the input systems in the training and validation datasets, including many combinations of these parameters that lie outside the boundary of the input domains described above. For experimental evaluation of the operator $\mathscr{R}$, systems characterized with input parameters outside the boundary of the ranges associated with the training and validation datasets are randomly selected from the testing dataset. These systems provide a more challenging task for the operator compared to the systems within the training ranges. The same process is followed for the other 3 one particle systems.

Figure 3 shows the errors incurred in the predictions of particle positions made by four $\mathscr{R}$ operators for 4 one particle 1D systems: a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential $U(x) = 1/2\ kx^2$ with $k = 12.8$, a particle of mass $m = 13$ and initial position $x_0 = -12.5$ in a double well potential $U(x) = x^4/4 - x^2/2$, a particle of mass $m = 13.2$ and initial position $x_0 = 3.4$ in a LJ potential $U(x) = 4\ \left(1/x^{12} - 1/x^6\right)$, and a particle of mass $m = 11.1$ and initial position $x_0 = -8.5$ in a rugged potential $U(x) = 1/50\left(x^4 - x^3 - 16x^2 + 4x + 48 + 10\sin\left(30x + 150\right)\right)$. Figures 3(B)–(E) show the errors incurred in the predictions of particle positions as a function of time t for timestep $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$ and 4000Δ respectively. These trajectory errors are computed as $\delta r(t) = |\vec{r}(t;\Delta_R) - \vec{r}_V(t,\Delta)|$, where $\vec{r}(t;\Delta_R)$ is the position vector of the particle at time t predicted by the RNN operator $\mathscr{R}$ with timestep Δ_R and $\vec{r}_V(t;\Delta)$ is the corresponding ground truth result produced by the Verlet integrator with timestep $\Delta = 0.001$. For $\Delta_R = 100\Delta$ (figure 3(B)), the errors $\delta r(t)$ are bounded between 0.001 and 0.0012 for a particle in a simple harmonic potential, a double well potential, and a LJ potential, while $\delta r(t) \in (0.0021, 0.0023)$ for a particle in a rugged potential. For all 4 one particle 1D systems, the errors increase as Δ_R increases from 100Δ to 4000Δ, but remain within the same order of magnitude. For example, for $\Delta_R = 4000\Delta$ (figure 3(E)), $\delta r(t) \in (0.0015, 0.0035)$ for a particle in a simple harmonic potential, a double well potential, and a LJ potential, and $\delta r(t) \in (0.003, 0.0045)$ for a particle in a rugged potential. For the same 4 1D systems, figure 3(A) shows the trajectory errors, $\delta r_V(t) = |\vec{r}_V(t;10\Delta) - \vec{r}_V(t;\Delta)|$, associated

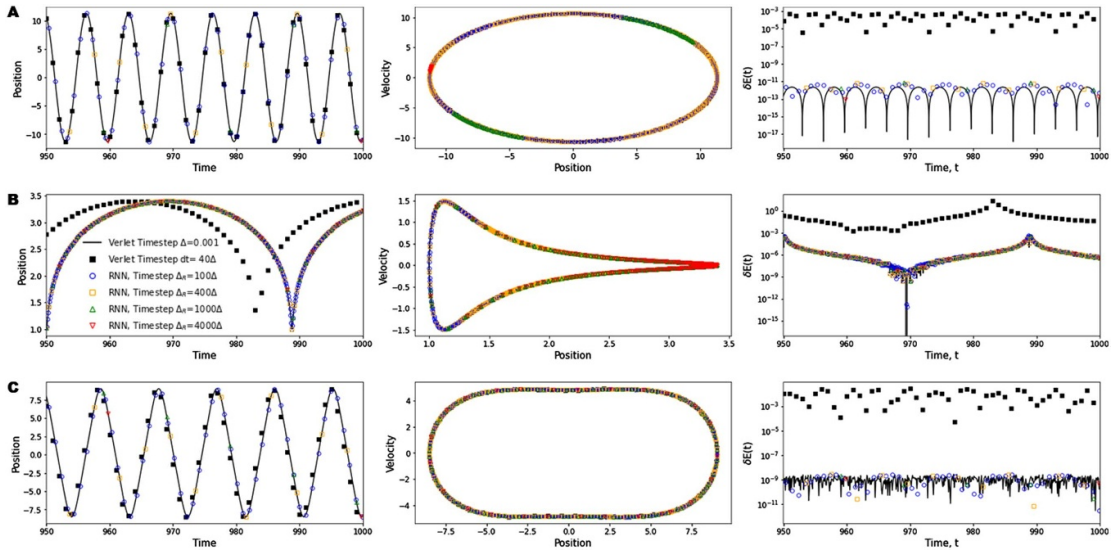

Figure 4. Time evolution performed by the RNN operators for a single particle in three 1D potentials from time $t = 950$ to 1000. Results are presented as position vs time (left column), velocity vs position (middle column), and energy deviation vs time (right column). Open symbols are dynamics predicted by the RNN operators with timestep $\Delta_R = 100\Delta$ (blue circles), 400Δ (orange squares), 1000Δ (green up triangles) and 4000Δ (red down triangles). Solid black squares are dynamics produced by the Verlet integrator with timestep 50Δ. (A) Dynamics of a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential with spring constant $k = 12.8$. (B) Dynamics of a particle of mass $m = 0.9$ and initial position $x_0 = 3.4$ in a Lennard–Jones potential. (C) Dynamics of a particle of mass $m = 8.0$ and initial position $x_0 = -8.5$ in a rugged potential.

with the time evolution performed using the Verlet integrator with a timestep of 10Δ. These trajectory errors are orders of magnitude larger compared to those incurred during the time evolution performed using the RNN operators, e.g. $\delta r_V \in (0.1, 0.5)$ for a particle in a rugged potential.

Errors in position predictions made by the $\mathscr{R}$ operators rise with increasing the complexity of the 1D potential. For example, for all Δ_R, trajectory errors are higher for the 1D system of a particle in a rugged potential compared to the 1D system of a particle in a simple harmonic potential. Figure 3(F) shows the errors in position predictions at time $t = 1000$ as a function of timestep for the same four 1D systems. In each case, the errors increase as the timestep Δ_R is increased from 100Δ to 4000Δ. However, the errors are bounded between 0.001 and 0.0045. On the other hand, the errors incurred in positions evolved using the Verlet integrator (figure 3(F) inset) rise exponentially with increasing timestep. For example, for a 10-fold increase in the timestep, the errors increase by four orders of magnitude for a particle in a LJ potential.

Figure 4 shows the predictions made by the RNN operators for positions, velocities, and energy deviations associated with the dynamics of a particle in three 1D potentials: particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential with $k = 12.8$ (A), particle of mass $m = 0.9$ and initial position $x_0 = 3.4$ in a LJ potential (B), and particle of mass $m = 8.0$ and initial position $x_0 = -8.5$ in a rugged potential (C). Results for each system are presented in three graphs: position vs time, velocity vs position, and energy deviation vs time. Energy deviation δE_t is defined as $\delta E_t = |E_t - E_0|/|E_0|$, where E_t and E_0 are the total energy of the system at time t and the initial time $t = 0$, respectively. By averaging across all times from $t = 0$ to $t = 1000$ and across all four 1D systems, the statistical errors in the positions predicted by $\mathscr{R}$ operators using timestep $\Delta_R = 100\Delta$, 400Δ, 1000Δ and 4000Δ are 0.00135, 0.00176, 0.00214 and 0.00297 respectively. Following the same process, the statistical errors in velocities predicted by $\mathscr{R}$ operators using timestep $\Delta_R = 100\Delta$, 400Δ, 1000Δ and 4000Δ are 0.00141, 0.00174, 0.00197 and 0.00293 respectively. The errors increase with increasing Δ_R, but remain within the same order of magnitude. The associated energy deviation δE_t tracks the ground truth energy deviation for all values of Δ_R. On the other hand, positions and velocities produced by the Verlet integrator with a timestep of 40Δ ($<\Delta_R$) show large deviations from the ground truth for all three 1D systems; the corresponding energy deviations are also orders of magnitude larger compared to the results obtained with RNN operators and the ground truth results.

In addition to the forward time evolution, we find that the RNN operators can accurately perform backward time evolution of 1D systems for an arbitrary length of time by utilizing the trajectory data in reverse order without undergoing any re-training using the time-reversed trajectories. Analogous to the process followed for the forward time evolution, we feed a sequence of length $S_R = 5$ of the future states of the trajectory starting at an arbitrary time $t + S_R\Delta_R$, and predict the state at time $t - \Delta_R$. The backward

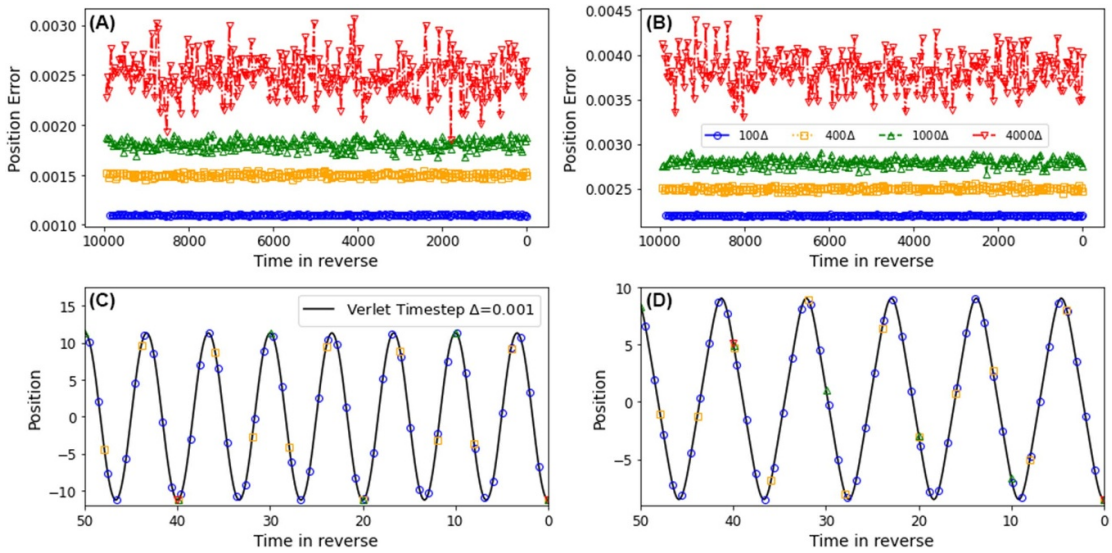

Figure 5. Trajectory errors and particle positions predicted during the backward time evolution performed by the RNN operators for two 1D systems: a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential with $k = 12.8$ ((A) and (C)), and a particle of mass $m = 11.1$ and initial position $x_0 = -8.5$ in a rugged potential ((B) and (D)). (A), (B) Errors incurred in positions predicted vs time t from $t = 10000$ to $t = 0$ for backward time evolution by the RNN operators with timestep $\Delta_R = 100\Delta$ (circles), 400Δ (squares), 1000Δ (up triangles), and 4000Δ (down triangles). (C), (D) Corresponding positions as a function of t for the two 1D systems from $t = 50$ to $t = 0$.

evolution terminates with the prediction at $t = 0$. Representative results for the backward time evolution are shown in figure 5 for a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential with $k = 12.8$ ((A) and (C)) and a particle of mass $m = 11.1$ and initial position $x_0 = -8.5$ in a rugged potential ((B) and (D)). Figures 5(A) and (B) show that the $\mathscr{R}$ operators generate accurate backward time evolution of these two systems starting from $t = 10000$ to $t = 0$ for $\Delta_R = 100\Delta, 400\Delta, 1000\Delta, 4000\Delta$. Errors in position predictions are bounded between 0.001 and 0.0045 for all Δ_R, similar to the errors in the forward trajectory evolution predicted by the same operators (figure 3). Figures 5(C) and (D) show the predicted positions vs time in reverse for the two systems from $t = 50$ to $t = 0$ for $\Delta_R = 100\Delta, 400\Delta, 1000\Delta, 4000\Delta$. In addition to exhibiting the time-reversal symmetry, we find that the RNN operators, with no explicit training to satisfy the symplectic condition, approximately preserve the symplectic property for timesteps up to 1000Δ (see appendix for details).

3.2. Few-particle systems in 3D

Our next set of experiments focus on training and testing the RNN-based $\mathscr{R}$ operators to predict the dynamics of few particles in 3D. Three separate $\mathscr{R}$ operators are designed to predict the dynamics of three few-particle systems with $N = 3$, $N = 8$, and $N = 16$ particles interacting via shifted and truncated LJ potentials in a cubic box with periodic boundary conditions. All particles have the same mass $m = 1$ and interact with the following LJ pair interaction potential:

$$U(r) = 4\epsilon\left(\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right) + 0.0163\,\epsilon \quad \text{for} \quad r \leqslant 2.5,$$
$$= 0 \quad \text{for} \quad r > 2.5.$$

For each of the three $N-$particle cases, the training and validation datasets consist of ground-truth trajectories produced by simulations of 5000 systems. These systems are generated by selecting different initial positions $\vec{r}_0$ for the N particles. The process begins by randomly selecting each of the three Cartesian coordinates x_0, y_0, z_0 of one particle between -3.0 and 3.0, and then placing all other particles next to the initial seed particle with a step size of 0.3508, ensuring that there are no particle overlaps and all particles have Cartesian coordinates between -3.0 and 3.0. In all simulations used to create the training and validation datasets, the initial velocities of all particles are set to 0, and the characteristic LJ energy ϵ is set to 1. Simulations are performed using the Verlet integrator with $\Delta = 0.001$ up to time $t = 2000$.

In each case, the entire dataset is randomly shuffled and separated into training and validation sets using a ratio of 0.8:0.2. For example, in the case of $N = 16$ particles system, 80% of the simulations (4000 systems) are randomly selected to form the training dataset, and the remaining 20% (1000 systems) are separated into

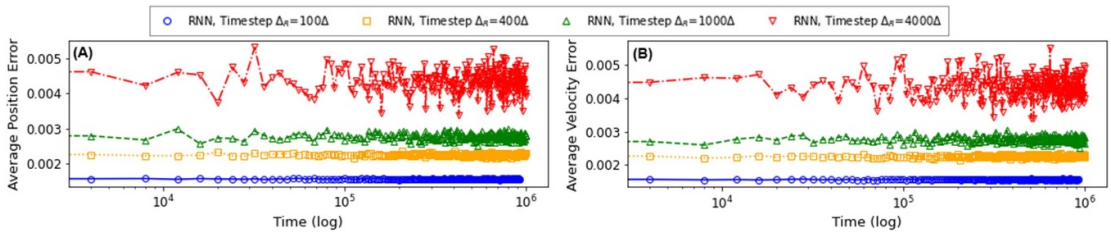

Figure 6. Average errors as a function of time t (log scale) associated with the predictions of the RNN operator for the positions (A) and velocities (B) of a 3D system of 16 particles interacting via Lennard–Jones forces under periodic boundary conditions. Results are shown for the time evolution performed using timestep $\Delta_R = 100\Delta$ (circles), 400Δ (squares), 1000Δ (up triangles) and 4000Δ (down triangles). For all Δ_R, the errors are $O(10^{-3})$ during the entire evolution up to $t = 10^6$.

the validation dataset. For experimental evaluation of the RNN operators, separate testing datasets for systems with $N = 3$, $N = 8$, and $N = 16$ particles are generated using simulations of particles of mass $m = 1$ performed up to time $t = 1000\,000$. In these simulations, the three Cartesian coordinates associated with the initial positions of all particles are randomly selected between -3.0 and 3.0, ensuring no overlapping particles. The initial velocities of particles are sampled from a Boltzmann distribution with a reduced temperature of 1, and the characteristic LJ energy ϵ is set to 2. Thus, the RNN operators are tasked to make predictions for test samples that are very different from the typical sample in the training and validation datasets (representative energy profiles associated with typical samples in test and training datasets are shown in figure 12 in appendix).

For all the three N-particle systems, we find that the associated $\mathscr{R}$ operators accurately evolve the positions and velocities of the particles with timestep Δ_R as large as 4000Δ, yielding energy-conserving dynamics up to time $t = 10^6$. In the interest of brevity, we discuss the results for the 3D system with $N = 16$ particles. Figures 6(A) and (B) show the average error associated with the RNN predictions for the positions and velocities of $N = 16$ particles as a function of time respectively. The average error in the prediction of positions is computed as $\delta r(t) = \sum_{i=1}^{N} |\vec{r}_i(t) - \vec{r}_{i,V}(t)|/N$, where $\vec{r}_i(t)$ is the 3D position vector of the i^{th} particle at time t predicted by the RNN operator $\mathscr{R}$ with timestep Δ_R and $\vec{r}_{i,V}(t)$ is the corresponding ground truth result at the same time t produced by the Verlet integrator with timestep $\Delta = 0.001$. The average error in the prediction of velocities is computed as $\delta v(t) = \sum_{i=1}^{N} |\vec{v}_i(t) - \vec{v}_{i,V}(t)|/N$, where $\vec{v}_i(t)$ is the 3D velocity vector of the i^{th} particle at time t predicted by $\mathscr{R}$ with timestep Δ_R and $\vec{v}_{i,V}(t)$ is the corresponding ground truth result at the same time t produced by the Verlet integrator with timestep $\Delta = 0.001$. Results are shown for time evolution performed by $\mathscr{R}$ for $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$, and 4000Δ. The errors $\delta r(t)$ and $\delta v(t)$ rise as Δ_R increases but remain bounded between 0.001 and 0.006 for time evolution up to $t = 10^6$.

Figure 7(A) shows the energy deviation $\delta E_t = |E_t - E_0|/|E_0|$ associated with the time evolution of the 3D system of 16 particles predicted by the RNN operator $\mathscr{R}$ using $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$, and 4000Δ. E_t and E_0 are the total energy of the system at time t and initial time ($t = 0$), respectively. For all values of Δ_R, the dynamics generated by the RNN operator are associated with $\delta E_t \lesssim 10^{-3}$ for up to $t = 10^6$ and track the ground truth result produced using the Verlet integrator with timestep Δ. In stark contrast, the dynamics produced using the Verlet integrator with a timestep of 40Δ (inset in figure 7(A)) exhibits a rapid energy divergence with $\delta E_t \sim 10^{12}$ for $t > 10^2$. Figure 7(B) shows the kinetic, potential, and total energies associated with the dynamics predicted by the RNN operator using timestep $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$, and 4000Δ. The corresponding ground truth results obtained with the Verlet integrator using timestep $\Delta = 0.001$ are also shown. For all Δ_R, the total energy predicted by $\mathscr{R}$ as a function of time is conserved. All the predicted energy profiles track the ground truth results up to $t = 10^6$.

3.3. Performance enhancement

We now discuss the performance enhancement resulting from using the deep learning approach presented here to perform simulations of one-particle and few-particle systems. For a given system, our approach uses the Verlet integrator to kickstart the simulation and the RNN operator $\mathscr{R}$ to evolve the dynamics forward in time. Incorporating this detail, we propose the following speedup metric S to quantify the performance enhancement:

$$S = \frac{S_T t_V}{S_V t_V + (S_T - S_V)\, t_R \Delta/\Delta_R}, \tag{5}$$

where S_T is the total number of steps needed if the time evolution is performed using only the Verlet integrator and $S_V = \Delta_R(S_R - 1)/\Delta$ is the total number of steps that generate the initial trajectories using

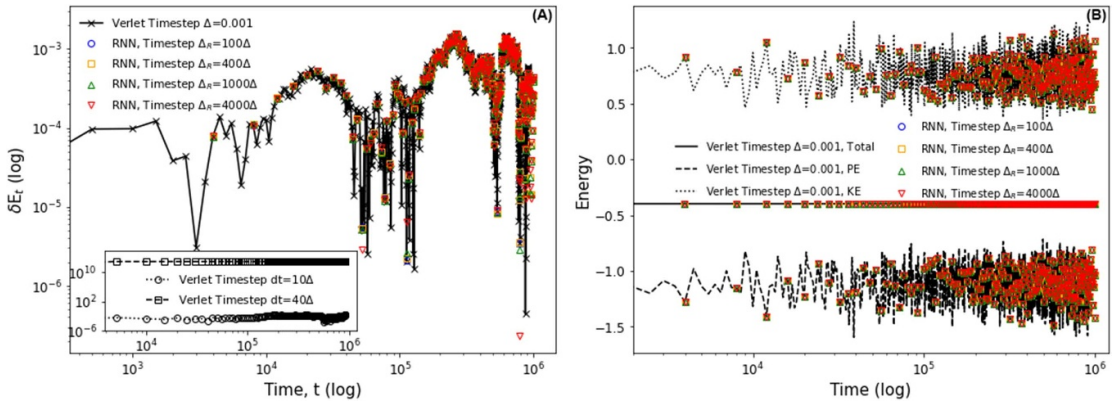

Figure 7. (A) Energy deviation δE_t (defined in the main text) as a function of time t (log scale) associated with the dynamics predicted by the RNN operator for the same 3D system of 16 particles as described in figure 6. Results are shown for the time evolution performed using timestep $\Delta_R = 100\Delta$ (circles), 400Δ (squares), 1000Δ (up triangles) and 4000Δ (down triangles). Inset shows the corresponding results using the Verlet integrator with timesteps 10Δ and 40Δ. For all values of Δ_R, the dynamics generated by the RNN operator track the ground truth result (black crosses) produced using the Verlet integrator with timestep $\Delta = 0.001$. (B) The total, potential, and kinetic energies associated with the dynamics predicted by the RNN operator using the same four Δ_R values; symbols have the same meaning as in (A). Lines represent the ground truth results obtained using the Verlet integrator for total (solid), potential (dashed), and kinetic (dotted) energies. For all Δ_R, the predicted energy profiles track the ground truth results.

Table 1. Speedup S for time evolution by the RNN operators using timestep Δ_R shown in the column heading.

Experiment	100Δ	200Δ	400Δ	1000Δ	2000Δ	4000Δ
1D, Simple harmonic	0.5	1.3	3.2	8.6	20.0	45.0
1D, Double well	0.6	1.2	2.8	8.7	17.3	38.0
1D, Lennard–Jones	0.9	1.5	3.9	12.8	22.5	42.3
1D, Rugged	0.4	0.8	2.1	4.7	9.7	20.6
3D, 8 particles	600	1000	1500	5500	8300	12 000
3D, 16 particles	3000	4900	7100	20 000	28 000	32 000

Verlet to kickstart the simulation. t_V and t_R are the times for one forward step propagation using Verlet and $\mathscr{R}$ respectively. In the speedup S, we have not accounted for the time spent on creating training and validation datasets, which is a one-time investment of <24 h for the experiments shown. S is 1 if $S_T = S_V$, that is, when no time evolution is performed using the RNN operator $\mathscr{R}$. In the limit $S_T \gg S_V$, we obtain $S \approx t_V \Delta_R/(t_R \Delta)$. Clearly, the greater the ratio Δ_R/Δ, the higher the speedup.

Table 1 shows the speedup S for the time evolution by the RNN operators using different timestep Δ_R. Results are shown for 1D experiments (first four rows) and 3D experiments (last two rows), and for $\Delta_R = 100$, 200, 400, 1000, 2000, and 4000. In all cases, S is computed for time evolution up to $t = 10^6$ with $S_T = 10^9$ steps. We find that the time t_V for one forward step propagation using Verlet varies by four orders of magnitude across the different experiments, ranging from $\approx 9 \times 10^{-6}$ s (for the 1D system with simple harmonic potential) to 4×10^{-2} s (for the 3D system with 16 particles). In contrast, the time t_R for one forward step propagation using the different RNN operators varies by only one order of magnitude across experiments, ranging from $\approx 3 \times 10^{-4}$ s (for the 1D system with simple harmonic potential) to $\approx 2 \times 10^{-3}$ s (for the 3D system with 16 particles).

We find that the speedup $S > 1$ for most experiments, signaling an enhancement in performance when the time evolution is performed using our deep learning approach. Low $S < 1$ values, observed mostly for the time evolution of the 1D systems with $\Delta_R = 100\Delta$, can be attributed to the relatively large time for one forward step propagation using RNN ($t_R \gg t_V$). As expected, S rises with increasing Δ_R. The largest values of S are recorded for 3D systems with more number of particles. In these cases, large increases in S result from both the use of large timestep Δ_R and the small time associated with the forward step propagation using RNN operators ($t_R < t_V$). For example, in the case of the time evolution of the 3D system of 16 particles with $\Delta_R = 4000\Delta$, we find that $t_R \approx 0.0026$ s is an order of magnitude smaller than $t_V \approx 0.04392$ s, resulting in the speedup $S \approx 32000$.

3.4. Limitations and outlook

We now explore the limits of the applicability of our deep learning approach. All RNN operators are trained using ground-truth trajectories associated with systems generated by sweeping input parameters over a finite

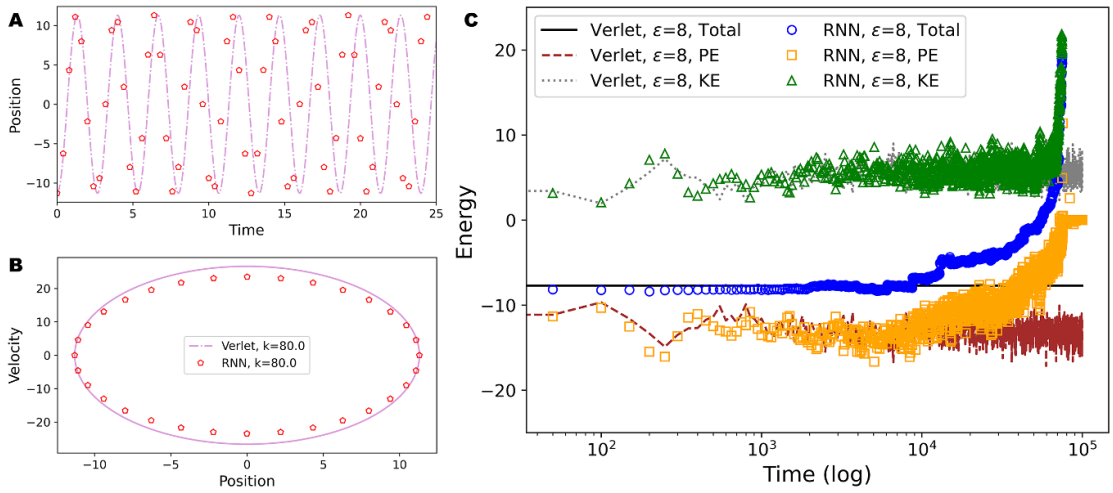

Figure 8. Failure cases illustrating the limits of the applicability of the deep learning approach. (A) and (B) show the time evolution predicted by the RNN operator with a timestep $\Delta_R = 100\Delta$ (open circles) for a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a 1D simple harmonic potential characterized with spring constant $k = 80.0$. Dynamics are shown as position vs time (A) and velocity vs position (B) plots from time $t = 0$ to $t = 25$. The RNN predictions quickly deviate from the ground truth results (lines) obtained using the Verlet integrator with timestep $\Delta = 0.001$. (C) shows the energies associated with the time evolution predicted by the RNN operator with a timestep $\Delta_R = 100\Delta$ (symbols) for the 3D system of 16 particles interacting via LJ potentials characterized with energy parameter $\epsilon = 8$. The total, potential, and kinetic energies are shown from time $t = 0$ to $t = 10^5$. All predicted energies start deviating quickly from the ground truth results (lines) obtained using the Verlet integrator with timestep $\Delta = 0.001$. The deviations get progressively larger with increasing time t.

range of values. Our results demonstrate that these operators can successfully perform time evolution of unseen input systems characterized with parameters that lie within and outside the ranges associated with the training datasets. However, as the input systems become very different from the systems in the training datasets, e.g. by selecting parameters that are well beyond the parameter ranges associated with the training datasets, we expect the RNN operators to produce inaccurate time evolution and to generate trajectories that deviate from the ground truth results.

Consider the 1D case of one particle in a simple harmonic potential for which we trained the RNN operator with ground-truth trajectories associated with input systems generated by sweeping over five discrete values of the initial position x_0 of the particle in the range $x_0 \in [-10, -2]$, 10 discrete values of particle mass m in the range $m \in [1, 10]$, and 10 discrete values of spring constant k in the range $k \in [1, 10]$. We extrapolated to an input system characterized with parameters $x_0 = -11.3, m = 14.4, k = 12.8$ in order to test the predictions of the RNN operator. For this test system, the operator produced accurate energy-conserving time evolution (section 3.1). However, for a particle with the same initial position $x_0 = -11.3$ and mass $m = 14.4$, the trajectories predicted by the RNN operator become progressively inaccurate as the spring constant k is increased to values greater than 2 times the maximum k value used in the training process (i.e. for $k \gtrsim 20$).

Figures 8(A) and (B) illustrate a failure case by showing the time evolution for a particle of mass $m = 14.4$ and initial position $x_0 = -11.3$ in a simple harmonic potential characterized with spring constant $k = 80.0$ (which is 8 times the maximum k value used in the training dataset). The time evolution by the RNN operator uses a timestep $\Delta_R = 100\Delta$, and results are shown from $t = 0$ to $t = 25$. After a small duration of time $t > 1$, the RNN predictions for positions and velocities deviate from the ground truth results obtained using the Verlet integrator. Recall that the same RNN operator predicted accurate time evolution up to $t = 1000$ for this 1D system with $k = 12.8$ (section 3.1). Similar limits in extrapolation and generalizability are observed for other 1D systems.

We next consider the 3D case of 16 particles interacting with LJ potentials in periodic boundary conditions. For this case, the RNN operator was trained with ground-truth trajectories associated with input systems generated by sweeping over many discrete values of the initial positions of the particles. Simulations of all systems in the training dataset were initialized with zero particle velocities and the LJ interactions between particles were characterized with energy parameter $\epsilon = 1$. To test the predictions of the RNN operator, we extrapolated to an input system for which the velocities of the 16 particles were sampled from a Boltzmann distribution with a reduced temperature of 1, and the LJ interactions were characterized with the energy parameter $\epsilon = 2$. For this test system, the operator produced accurate energy-conserving time evolution (section 3.2). However, we find that the time evolution predicted by the RNN operator becomes progressively inaccurate as ϵ is increased to values over 5.

Figure 8(C) illustrates a failure case by showing the energy profiles associated with the time evolution performed by the RNN operator for 16 particles interacting via LJ potentials characterized with $\epsilon = 8$. Simulation is initialized with randomly selected positions and with velocities sampled from the Boltzmann distribution at a reduced temperature of 1. The time evolution by the RNN operator uses a timestep $\Delta_R = 100\Delta$, and results are shown from $t = 0$ to $t = 10^5$. All energies (kinetic, potential, and total) start deviating from the ground truth results right from the beginning, the deviations getting progressively stronger with increasing time t. As t increases to values beyond 10^4, the total energy starts to diverge. Recall that the same RNN operator predicted accurate time evolution from $t = 0$ to $t = 10^6$ for this 3D system with $\epsilon = 2$ (section 3.2). For both one particle systems in 1D and few particle systems in 3D, addressing the failure cases will involve expanding the range of input parameters characterizing the systems and including the associated trajectory data in the training of the RNN operators.

In our current formulation, the RNN operators are designed and trained to furnish the time evolution of systems characterized by the selected potential energy describing the particles. Thus, the RNN operator trained to learn the dynamics of one potential energy landscape (e.g. one particle in a simple harmonic potential) is, by design, not capable to predict the dynamics of another closely related but qualitatively different potential energy landscape (e.g. one particle in a double well potential). For a complex potential energy landscape with multiple basins and barriers, the associated RNN operator will need to 'see' a diverse group of trajectories corresponding to different regions of the energy landscape in order to accurately furnish the time evolution. The complexity of the energy landscape may require changes in the architectural configuration of the RNN operators and may also lead to longer training times. Similarly, the RNN operators will need to be trained with ground truth trajectories associated with different representative assembly behaviors (e.g. phase changes in 3D systems of many particles interacting with LJ potentials) in order for them to successfully evolve the dynamics for corresponding thermodynamic statepoints.

Our future work will explore ways to enhance the generalizability of the RNN operators and to extend the applicability of our deep learning approach to systems described with complex energy landscapes. In this initial study, we have limited our focus on few-particle systems and on a single type of thermodynamic ensemble. Future work will explore the scaling of the approach to a larger number of particles and will examine the accuracy of the RNN operators in different thermodynamic ensembles.

4. Conclusion

We have introduced a deep learning approach that utilizes RNNs to design operators that solve Newton's equations of motion and evolve the dynamics of particles by utilizing timesteps orders of magnitude larger than the typical timestep used in numerical integrators such as Verlet. We have obtained state-of-the-art results in terms of the timesteps, the number of particles, and the complexity of the potential characterizing the interactions between particles. The RNN operators learn both the interaction potentials and the dynamics of the particles based on their experience with the ground-truth solutions of Newton's equations of motion. These operators produce accurate predictions for the time evolution of particles accompanied with excellent energy conservation over a variety of force fields using up to 4000× larger timestep than the Verlet integrator. The use of deep learning methods in tasks central to MD simulations is a critical first step towards the long-term goal of machine-learning-assisted MD simulations of many-particle systems. Further, the idea of formulating the dynamics of particles into a sequence processing problem solved via the use of RNNs illustrates an important approach to learn the time evolution operators, which is applicable across different fields including fluid dynamics and robotics [31, 33, 50].

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

This work is partially supported by the NSF through Awards 1720625 and DMR-1753182, and by the DOE through Award DE-SC0021418. G C F was partially supported by NSF CIF21 DIBBS 1443054 and CINES 1835598 Awards. V J thanks M O Robbins for useful discussions.

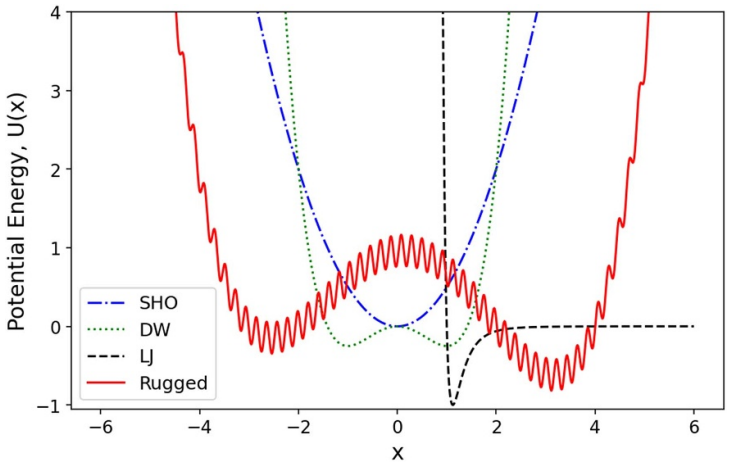

Figure 9. Potential energies associated with the 1D experiments. Dash-dotted, dotted, dashed, and solid lines represent simple harmonic (SHO), double well (DW), Lennard–Jones (LJ), and rugged potentials respectively.

Appendix

Training and validation dataset preparation for 1D systems

Here, we describe the datasets used for training the RNN operators to predict the dynamics of 1D one-particle systems. The potential energy functions characterizing the four 1D systems are shown in figure 9. In each case, the Verlet integrator with a timestep $\Delta = 0.001$ is used to generate the ground truth trajectories for up to time $t = 200$. For all four 1D systems described below, the testing dataset to evaluate the predictions of the associated RNN-based operator comprises 100 input systems characterized with parameters distinct from those used in the input systems in the training and validation datasets. These 100 test systems also include systems characterized with all parameters lying outside the boundary of the parameter ranges associated with systems in the training and validation datasets.

Particle in a simple harmonic potential
For this system, the potential energy is given by:

$$U = \frac{1}{2}kx^2, \tag{6}$$

where k is the spring constant. The training and validation datasets consist of ground-truth trajectories associated with simulations of 500 input systems generated by sweeping over 5 discrete values of initial position of the particle $x_0 = -10, -8, -6, -4, -2$; 10 discrete values of particle mass $m = 1, 2, \ldots, 9, 10$; and 10 discrete values of spring constant $k = 1, 2, \ldots, 9, 10$. Each simulation produces a trajectory with 400 000 position and velocity values.

Particle in a double well potential
For this system, the potential energy is given by:

$$U = \frac{1}{4}x^4 - \frac{1}{2}x^2. \tag{7}$$

The training and validation datasets consist of ground-truth trajectories associated with simulations of 500 input systems generated by sweeping over 10 discrete values of particle mass $m = 1, 2, \ldots, 9, 10$; and 50 uniformly-distributed discrete values of initial position of the particle $x_0 \in [-3.1, 3.1]$. Each simulation produces a trajectory with 400 000 position and velocity values.

Particle in a Lennard–Jones potential
For this system, the potential energy is given by:

$$U(x) = 4\left(\left(\frac{1}{x}\right)^{12} - \left(\frac{1}{x}\right)^{6}\right). \tag{8}$$

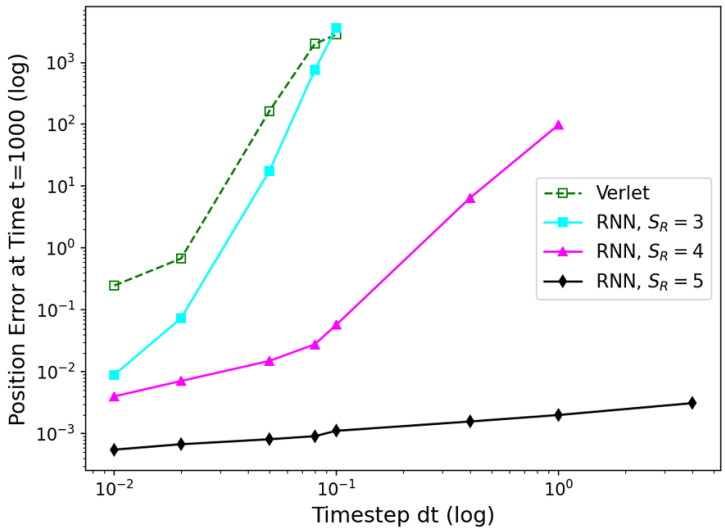

Figure 10. Error δr as a function of timestep dt incurred in updating the position of the particle at time $t = 1000$ for a 1D system of a particle of mass $m = 1$ in a Lennard–Jones potential with initial position $x_0 = 2.0$. δr is evaluated by comparing the predictions to the ground truth results obtained using the Verlet integrator with a small timestep $\Delta = 0.001$. Closed squares, triangles, and diamonds are errors incurred when using RNN operators trained with sequence length $S_{\mathrm{R}} = 3, 4$ and 5 respectively. Open squares correspond to the errors produced by the Verlet integrator.

The training and validation datasets consist of ground-truth trajectories associated with simulations of 500 input systems generated by sweeping over 10 discrete values of particle mass $m = 1, 2, \ldots, 9, 10$; and 50 uniformly-distributed discrete values of initial position of the particle $x_0 \in [1.0, 3.0]$. Each simulation produces a trajectory with 400 000 position and velocity values.

Particle in a rugged potential

For this system, the potential energy [15] is given by:

$$U(x) = \frac{x^4 - x^3 - 16x^2 + 4x + 48}{50} + \frac{\sin(30(x+5))}{5}. \tag{9}$$

The training and validation datasets consist of ground-truth trajectories associated with simulations of 640 input systems generated by sweeping over 10 discrete values of particle mass $m = 1, 2, \ldots, 9, 10$; and 64 uniformly-distributed discrete values of initial position of the particle $x_0 \in [-6.1, 6.1]$. Each simulation produces a trajectory with 400 000 position and velocity values.

Training RNN operators with different sequence lengths

The sequence length S_{R} is defined as the number of past configurations used by the RNN operator to predict the future configuration. Using the training, validation and testing datasets associated with the 1D system of one particle in a LJ potential, we did experiments to compare the accuracy of RNN operators designed using $S_{\mathrm{R}} = 3, 4$ and 5. Figure 10 shows the error δr in the prediction of the position of the particle at time $t = 1000$ for a system in the test dataset as a function of the timestep dt. δr is evaluated by comparing the RNN predictions to the ground truth results obtained using the Verlet integrator with a small timestep $\Delta = 0.001$. We find that the RNN operator trained with $S_{\mathrm{R}} = 3$ produces errors that rise steeply from 0.0089 to 3640.75, spanning over 5 orders of magnitude, as timestep dt is increased from 10Δ to 100Δ. The rise in these errors is similar to the increase observed for position errors incurred using the Verlet integrator with increasing dt. The accuracy improves and the errors are comparatively reduced for the RNN operator trained with $S_{\mathrm{R}} = 4$. The errors are now bounded between 0.004 and 0.058 for $dt \lesssim 100\Delta$ but quickly rise to 97.89 for $dt = 1000\Delta$. In stark contrast, the RNN operator trained with sequence length $S_{\mathrm{R}} = 5$ produces errors that exhibit a much weaker rise from 0.0005 to 0.003 as the timestep dt is increased from 100Δ to 4000Δ.

Symplectic property

In the main text, we showed that the RNN operators exhibited time-reversal symmetry. In this subsection, we explore numerically whether the trajectories predicted by the RNN operators preserve the symplectic

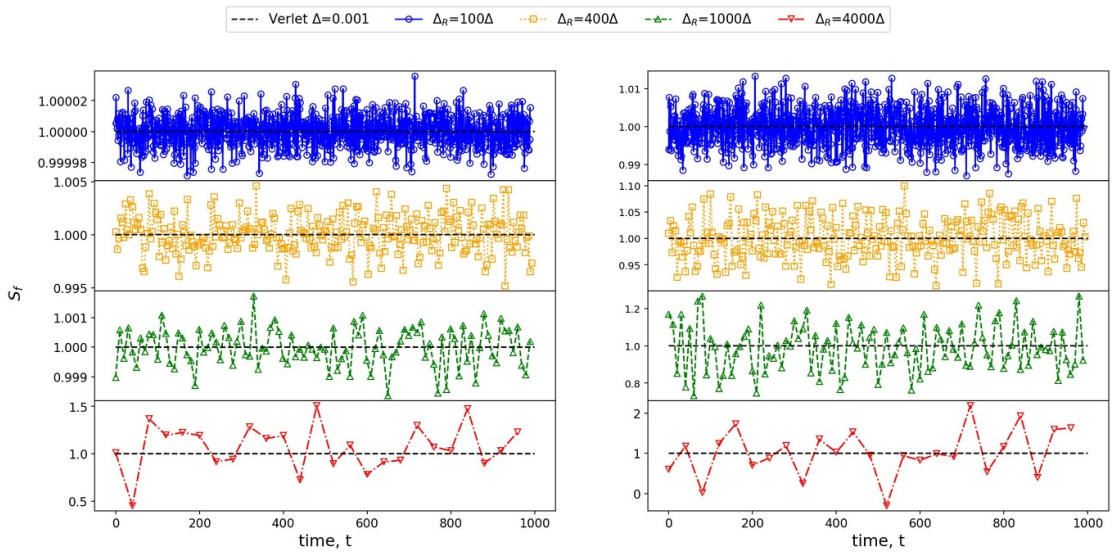

Figure 11. Evaluating the preservation of the symplectic property by the RNN operators for 1D systems by computing S_f (defined in equation (10) of the main text) from time $t = 0$ to $t = 1000$. Results are shown for a particle in a simple harmonic potential with parameters $m = 10$, $k = 1$, $x_0 = -10$ (left column), and for a particle in a rugged potential with parameters $m = 8$, $x_0 = -8$ (right column). The trajectories are predicted using RNN operators with timestep $\Delta_R = 100\Delta$ (circles), 400Δ (squares), 1000Δ (triangles), and 4000Δ (pentagons). Black lines represent Verlet results with timestep $\Delta = 0.001$. While the Verlet integrator maintains $S_f = 1$ (up to numerical precision), the RNN operators approximately satisfy the symplectic condition ($S_f \approx 1$) up to 1000Δ. Deviations from 1 become much larger for the highest timestep of 4000Δ. For a given timestep, fluctuations of S_f near 1 are also larger for the more complex 1D potential (rugged).

property. We note that these operators are not trained explicitly to preserve the symplectic structure. For the sake of simplicity, we focus the investigation on 1D systems. In these cases, the symplectic property is obeyed if the trajectories generated using the RNN operators satisfy the equality:

$$JMJ^T = M, \tag{10}$$

where

$$J = \begin{bmatrix} \frac{\partial \vec{x}(t)}{\partial \vec{x}(0)} & \frac{\partial \vec{x}(t)}{\partial \vec{p}(0)} \\ \\ \frac{\partial \vec{p}(t)}{\partial \vec{x}(0)} & \frac{\partial \vec{p}(t)}{\partial \vec{p}(0)} \end{bmatrix},$$

is the Jacobian matrix. Here $\vec{x}(t), \vec{p}(t)$ are the positions and momenta associated with the trajectory of the particles at time t, and $\vec{x}(0), \vec{p}(0)$ are the initial positions and momenta at $t = 0$. The matrix M is given by:

$$M = \begin{bmatrix} \vec{0} & \vec{I} \\ -\vec{I} & \vec{0} \end{bmatrix},$$

where 0 and I are $dN \times dN$ dimensional zero and identity matrices, respectively (d is the spatial dimension and N is the number of particles). For the case of one particle in 1D, $d = 1$ and $N = 1$, and the matrix M becomes:

$$M = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

The left hand side of equation (10) can be simplified as:

$$JMJ^T = \begin{bmatrix} 0 & S_f \\ -S_f & 0 \end{bmatrix}$$

where S_f is given by:

$$S_f = \frac{\partial \vec{x}(t)}{\partial \vec{x}(0)} \frac{\partial \vec{p}(t)}{\partial \vec{p}(0)} - \frac{\partial \vec{x}(t)}{\partial \vec{p}(0)} \frac{\partial \vec{p}(t)}{\partial \vec{x}(0)}. \tag{11}$$

Using the symplectic condition (equation (10)), we find:

$$\begin{bmatrix} 0 & S_f \\ -S_f & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Thus, the RNN operators satisfy the symplectic property when:

$$S_f = 1. \tag{12}$$

We computed S_f for the four 1D systems using equation (11) and the trajectory data predicted by the associated RNN operators with different timestep Δ_R. The ground truth S_f values were obtained using the Verlet integrator with timestep $\Delta = 0.001$. We find that for all systems, while the Verlet integrator yields $S_f = 1$ up to the numerical precision, the RNN operators approximately satisfy the symplectic condition. S_f fluctuates around 1, with the extent of fluctuations becoming stronger with increasing Δ_R and potential complexity. Figure 11 shows representative results for a particle in a simple harmonic potential with parameters $m = 10$, $k = 1$, $x_0 = -10$, and for a particle in a rugged potential with parameters $m = 8$, $x_0 = -8$. In the case of the particle in a simple harmonic potential, S_f exhibits small fluctuations with a mean of 1.0 and standard deviation (fractional error) of $\approx$0.001 for timesteps $\Delta_R = 100\Delta, 400\Delta, 1000\Delta$. However, when Δ_R is increased to 4000Δ, S_f exhibits greater fluctuations with a mean of 1.0 and standard deviation of $\approx$0.5. Similar trends are observed for the particle in a rugged potential, albeit with greater fluctuations in S_f around 1 for each Δ_R, which can be attributed to the greater complexity of the rugged potential.

Energy profiles for representative samples of few-particle systems in training and testing datasets

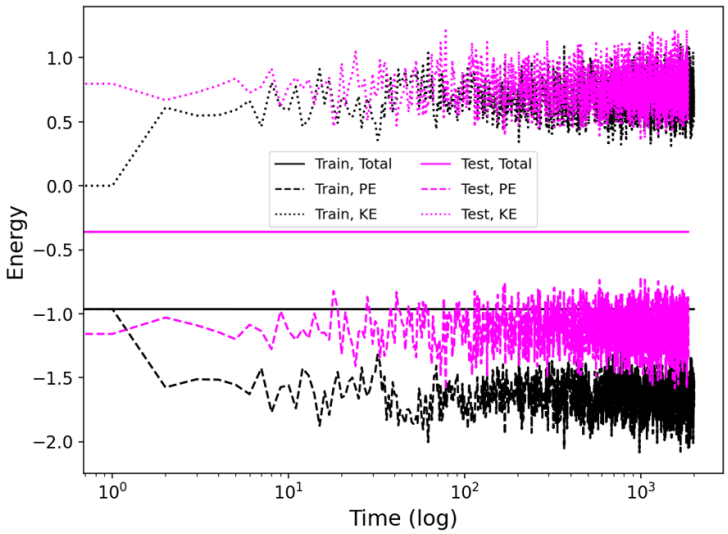

Figure 12. The total, potential, and kinetic energies vs time (log scale) associated with a representative system in the training dataset (black lines) compared with the energy profiles for the test system (magenta lines) of 16 particles in 3D interacting with Lennard–Jones forces, as described in figure 6 of the main text. For the system in the training dataset, the initial positions are randomly selected, the initial velocities are set to 0, and the Lennard–Jones energy parameter $\epsilon = 1$. For the test system, the initial positions are randomly selected, the initial velocities are sampled from a Boltzmann distribution with a reduced temperature of 1, and $\epsilon = 2$. Both sets of energy profiles represent ground truth results obtained using the Verlet integrator with timestep $\Delta = 0.001$.

ORCID iD

Vikram Jadhao https://orcid.org/0000-0002-8034-2654

References

[1] Newton I 1687 *Philosophiae Naturalis Principia Mathematica* (London: Royal Society)
[2] Alder B J and Wainwright T E 1959 Studies in molecular dynamics. I. General method *J. Chem. Phys.* **31** 459–66
[3] Frenkel D and Smit B 2001 *Understanding Molecular Simulation* 2nd edn (New York: Academic)
[4] Verlet L 1967 Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules *Phys. Rev.* **159** 98
[5] Andersen H C 1983 Rattle: a 'velocity' version of the shake algorithm for molecular dynamics calculations *J. Comput. Phys.* **52** 24–34
[6] Butcher J C 2016 *Numerical Methods for Ordinary Differential Equations* (New York: Wiley)
[7] Wu Y *et al* 2016 Google's neural machine translation system: bridging the gap between human and machine translation (arXiv:1609.08144)
[8] Chong E, Han C and Park F C 2017 Deep learning networks for stock market analysis and prediction: methodology, data representations and case studies *Expert Syst. Appl.* **83** 187–205
[9] Huang X, Fox G C, Serebryakov S, Mohan A, Morkisz P and Dutta D 2019 Benchmarking deep learning for time series: challenges and directions *2019 IEEE Int. Conf. Big Data (Big Data)* (IEEE) pp 5679–82
[10] Ferguson A L 2017 Machine learning and data science in soft materials engineering *J. Phys.: Condens. Matter* **30** 043002
[11] Butler K T, Davies D W, Cartwright H, Isayev O and Walsh A 2018 Machine learning for molecular and materials science *Nature* **559** 547
[12] Fox G *et al* 2019 Learning everywhere: pervasive machine learning for effective high-performance computation *HPDC Workshop at IPDPS 2019*
[13] Guo A Z, Sevgen E, Sidky H, Whitmer J K, Hubbell J A and de Pablo J J 2018 Adaptive enhanced sampling by force-biasing using neural networks *J. Chem. Phys.* **148** 134108
[14] Botu V and Ramprasad R 2015 Adaptive machine learning framework to accelerate *ab initio* molecular dynamics *Int. J. Quantum Chem.* **115** 1074–83
[15] Wang J, Olsson S, Wehmeyer C, Pérez A, Charron N E, De Fabritiis G, Noé F and Clementi C 2019 Machine learning of coarse-grained molecular dynamics force fields *ACS Cent. Sci.* **5** 755–67
[16] Kadupitiya J C S, Fox G C and Jadhao V 2020 Machine learning for parameter auto-tuning in molecular dynamics simulations: efficient dynamics of ions near polarizable nanoparticles *Int. J. High Perform. Comput. Appl.* **34** 357–74
[17] Long A W, Zhang J, Granick S and Ferguson A L 2015 Machine learning assembly landscapes from particle tracking data *Soft Matter* **11** 8141–53
[18] Spellings M and Glotzer S C 2018 Machine learning for crystal identification and discovery *AIChE J.* **64** 2198–206

[19] Sharp T A, Thomas S L, Cubuk E D, Schoenholz S S, Srolovitz D J and Liu A J 2018 Machine learning determination of atomic dynamics at grain boundaries *Proc. Natl Acad. Sci.* **115** 10943–7
[20] Moradzadeh A and Aluru N R 2019 Molecular dynamics properties without the full trajectory: a denoising autoencoder network for properties of simple liquids *J. Phys. Chem. Lett.* **10** 7568–76
[21] Sun Y, DeJaco R F and Siepmann J I 2019 Deep neural network learning of complex binary sorption equilibria from molecular simulation data *Chem. Sci.* **10** 4377–88
[22] Häse F, Galván I F, Aspuru-Guzik A, Lindh R and Vacher M 2019 How machine learning can assist the interpretation of *ab initio* molecular dynamics simulations and conceptual understanding of chemistry *Chem. Sci.* **10** 2298–307
[23] Kadupitiya J C S, Fox G C and Jadhao V 2019 Machine learning for performance enhancement of molecular dynamics simulations *Int. Conf. Computational Science* pp 116–30
[24] Kadupitiya J C S, Sun F, Fox G and Jadhao V 2020 Machine learning surrogates for molecular dynamics simulations of soft materials *J. Comput. Sci.* **42** 101107
[25] Raissi M and Karniadakis G E 2018 Hidden physics models: machine learning of nonlinear partial differential equations *J. Comput. Phys.* **357** 125–41
[26] Long Z, Lu Y, Ma X and Dong B 2018 PDE-Net: learning PDEs from data *Int. Conf. Machine Learning* (PMLR) pp 3208–16
[27] Chen T Q, Rubanova Y, Bettencourt J and Duvenaud D K 2018 Neural ordinary differential equations *Advances in Neural Information Processing Systems* pp 6571–83
[28] Endo K, Tomobe K and Yasuoka K 2018 Multi-step time series generator for molecular dynamics *32nd AAAI Conf. Artificial Intelligence*
[29] Breen P G, Foley C N, Boekholt T and Zwart S P 2020 Newton versus the machine: solving the chaotic three-body problem using deep neural networks *Mon. Not. R. Astron. Soc.* **494** 2465–70
[30] Chen Z, Zhang J, Arjovsky M and Bottou L 2019 Symplectic recurrent neural networks (arXiv:1909.13334)
[31] Shen P, Zhang X and Fang Y 2017 Essential properties of numerical integration for time-optimal path-constrained trajectory planning *IEEE Robot. Autom. Lett.* **2** 888–95
[32] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[33] Bar-Sinai Y, Hoyer S, Hickey J and Brenner M P 2019 Learning data-driven discretizations for partial differential equations *Proc. Natl Acad. Sci.* **116** 15344–9
[34] Shen X, Cheng X and Liang K 2020 Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method (arXiv:2003.09573)
[35] Raissi M, Perdikaris P and Karniadakis G E 2018 Multistep neural networks for data-driven discovery of nonlinear dynamical systems (arXiv:1801.01236)
[36] Tsai S-T, Kuo E-J and Tiwary P 2020 Learning molecular dynamics with simple language model built upon long short-term memory neural network *Nat. Commun.* **11** 1–11
[37] Greydanus S, Dzamba M and Yosinski J 2019 Hamiltonian neural networks *Advances in Neural Information Processing Systems* vol 32
[38] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D and Ho S 2020 Lagrangian neural networks (arXiv:2003.04630)
[39] Chmiela S, Tkatchenko A, Sauceda H E, Poltavsky I, Schütt K T and Müller K-R 2017 Machine learning of accurate energy-conserving molecular force fields *Sci. Adv.* **3** e1603015
[40] Sanchez-Gonzalez A, Bapst V, Cranmer K and Battaglia P 2019 Hamiltonian graph networks with ODE integrators (arXiv:1909.12790)
[41] Minary P, Tuckerman M E and Martyna G J 2004 Long time molecular dynamics for enhanced conformational sampling in biomolecular systems *Phys. Rev. Lett.* **93** 150201
[42] Morrone J A, Markland T E, Ceriotti M and Berne B J 2011 Efficient multiple time scale molecular dynamics: using colored noise thermostats to stabilize resonances *J. Chem. Phys.* **134** 014103
[43] Leimkuhler B, Margul D T and Tuckerman M E 2013 Stochastic, resonance-free multiple time-step algorithm for molecular dynamics with very large time steps *Mol. Phys.* **111** 3579–94
[44] Chen P-Y and Tuckerman M E 2018 Molecular dynamics based enhanced sampling of collective variables with very large time steps *J. Chem. Phys.* **148** 024106
[45] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80
[46] Chollet F *et al* 2018 Keras: The Python Deep Learning library [ascl:1806.022] (https://github.com/keras-team/keras)
[47] Buitinck L *et al* 2013 API design for machine learning software: experiences from the scikit-learn project (arXiv:1309.0238)
[48] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics* pp 249–56
[49] GitHub 2021 Repository RNN-MD in softmaterialslab (available at: https://github.com/softmaterialslab/RNN-MD/)
[50] Kates-Harbeck J, Svyatkovskiy A and Tang W 2019 Predicting disruptive instabilities in controlled fusion plasmas through deep learning *Nature* **568** 526–31