Learning Large-Time-Step Molecular Dynamics with Graph Neural Networks

Tianze Zheng
Tsinghua University
Beijing, China

Weihao Gao
ByteDance Inc.
Beijing, China

Chong Wang
ByteDance Inc.
Beijing, China
chong.wang@bytedance.com

Abstract

Molecular dynamics (MD) simulation predicts the trajectory of atoms by solving Newton's equation of motion with a numeric integrator. Due to physical constraints, the time step of the integrator need to be small to maintain sufficient precision. This limits the efficiency of simulation. To this end, we introduce a graph neural network (GNN) based model, MDNet, to predict the evolution of coordinates and momentum with large time steps. In addition, MDNet can easily scale to a larger system, due to its linear complexity with respect to the system size. We demonstrate the performance of MDNet on a 4000-atom system with large time steps, and show that MDNet can predict good equilibrium and transport properties, well aligned with standard MD simulations.

1 Introduction

Molecular dynamics (MD) is a simulation technique for analyzing the movement of many-body systems by solving Newton's equation of motion. The resulting trajectories provide a view of dynamic evolution of atoms and molecules, and can be used to analyze the equilibrium and transport properties of the system [1]. MD plays a key role in many scientific fields, including materials science [2, 3], biochemistry [4], drug discovery [5], etc.

The core of an MD algorithm is to solve Newton's equation of motion at discrete time steps. One of the most frequently used algorithm is the ordinary Verlet integrator [6], which predicts the coordinate of atom i at time t, $\mathbf{q}_i(t)$, after time step Δt by:

$$\mathbf{q}_i(t+\Delta t) = 2\mathbf{q}_i(t) - \mathbf{q}_i(t-\Delta t) - \frac{\partial \mathcal{U}}{\partial \mathbf{q}_i(t)}\frac{\Delta t^2}{m_i} + \mathcal{O}(\Delta t^4), \tag{1}$$

where the potential energy $\mathcal{U}$ of the system is a function of the coordinates $\mathbf{q}$. The time step Δt must be set small enough to avoid the contribution of $\mathcal{O}(\Delta t^4)$. In practice, Δt is usually chosen at femtosecond scale (1 fs = 10^{-15}s). Recently, efforts have been made to solve the classical mechanics of many-body systems via machine learning methods [7–12]. However, all these methods are limited to small systems containing only dozens of atoms, whereas practical MD simulations usually involve thousands to millions of atoms. Therefore, a scalable machine learning approach with linear complexity is highly desired.

In this paper, we propose MDNet, a scalable graph neural network (GNN) model to predict the dynamics of many-body systems. The key contribution is to predict the movement of an atom based

NeurIPS 2021 AI for Science Workshop.

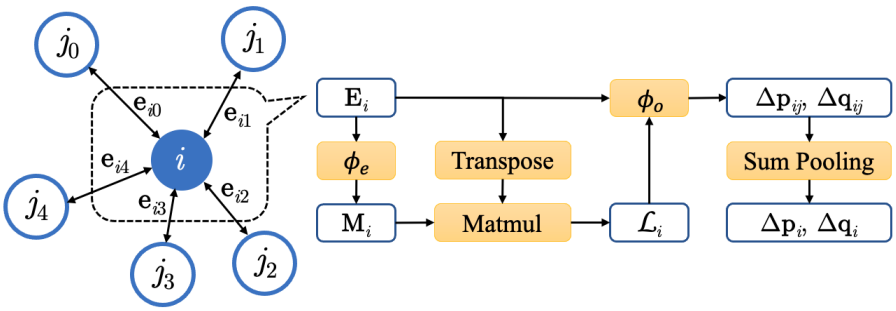

Figure 1: Illustration of model architecture of MDNet.

on its local environment encoded by a series of descriptors similar as DeePMD [13, 14]. In addition, we introduce momenta in the local environment descriptor and make the model comply with the law of conservation of momenta. We show that MDNet can simulate a system containing thousands of atoms with a time step 128 times larger than that of the Verlet integrator. The resulting equilibrium and transport properties are shown to be consistent with the simulations using the Verlet integrator.

1.1 Related Work

The trajectories of MD simulations can be viewed as sequence data hence could be learned by sequence models, such as recurrent neural networks (RNNs). Tsai et al. showed that one-dimensional stochastic trajectories generated from higher-dimensional dynamics can be learned by long short-term memory network (LSTM). Kadupitiya et al. demonstrated that LSTM was able to predict MD trajectory for systems containing 16 atoms. Another type of models including HNN [10], SRNN [11], GFNN [12] and others [15–19], manged to incorporate physical knowledge and solve mechanical problems by learning Hamiltonian or other physical quantities. However, all the aforementioned models used the information of all atoms to predict the dynamics, which limits their scalability.

In traditional MD simulations, a common assumption is that atoms only interact with their neighbors. Motivated by this assumption, Satorras et al. proposed E(n) equivariant graph neural networks (EGNNs), which model the system by a graph and achieve linear complexity [20]. Compared to EGNN, MDNet use more sophisticated descriptors of local environments and achieve better performance.

2 Background

We briefly introduce some basic notations in Hamiltonian mechanics that MDNet is built upon. The Hamiltonian for an N-particle system in d dimensions can be written as:

$$\mathcal{H}(\mathbf{p},\mathbf{q}) = \mathcal{T}(\mathbf{p}) + \mathcal{U}(\mathbf{q}), \quad \mathbf{p},\mathbf{q} \in \mathbb{R}^{N\times d}, \tag{2}$$

where $\mathbf{p},\mathbf{q}$ are the momenta and coordinates of the atoms. The momentum is defined as $\mathbf{p}_i = m_i\mathbf{v_i}$, where m_i and $\mathbf{v}$ are the mass and velocity of atom i, respectively. $\mathcal{T}(\mathbf{p}) = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i}$ is the kinetic energy of the system and $\mathcal{U}(\mathbf{q})$ is the potential energy induced by interaction between atoms. The motion of atoms is determined by Hamilton's equation:

$$\dot{\mathbf{q}}_\mathbf{i} = \frac{\partial\mathcal{H}}{\partial\mathbf{p}_i} = \frac{\mathbf{p}_i}{m_i}, \quad \dot{\mathbf{p}}_\mathbf{i} = -\frac{\partial\mathcal{H}}{\partial\mathbf{q}_i} = -\frac{\partial\mathcal{U}}{\partial\mathbf{q}_i} = \mathbf{F}_i. \tag{3}$$

which is a generalization of Newton's equation of motion. The equation is then solved at discrete time steps via numeric integrators, where the velocity Verlet (appendix A) [21] is the most widely used one.

3 Model Architecture

In Hamilton's equation (3), the variation of $\mathbf{q}$ only depends on $\mathbf{p}$ and vice versa. However, with a larger time step, atoms interact with each other, thus, $\Delta\mathbf{q}$ depends not only on $\mathbf{p}$, but also $\mathbf{q}$ (and so is

$\Delta\mathbf{p}$). In MDNet, we model $\Delta\mathbf{q}$ and $\Delta\mathbf{p}$ via neural networks:

$$\Delta\mathbf{q}, \Delta\mathbf{p} = \mathrm{MDNet}\left[\mathbf{q}(t), \mathbf{p}(t), \Delta t\right], \tag{4}$$

and the integration algorithm becomes:

$$\mathbf{p}(t+\Delta t) = \mathbf{p}(t) + \Delta\mathbf{p},\ \mathbf{q}(t+\Delta t) = \mathbf{q}(t) + \Delta\mathbf{q}. \tag{5}$$

In MDNet, the system is represented by a graph where each atom is represented by a node. Two nodes are connected by an edge if the distance between them is less than a cut-off distance, r_c. For an edge between node i and j, its edge feature and message is calculated as:

$$\mathbf{e}_{ij} = [\frac{1}{r_{ij}}, \frac{\mathbf{q}_i - \mathbf{q}_j}{r_{ij}^2}, \mathbf{p}_i - \mathbf{p}_j]^T,\ \ \mathbf{m}_{ij} = \phi_e(\mathbf{e}_{ij}) \tag{6}$$

where $r_{ij} = ||\mathbf{q}_i - \mathbf{q}_j||_2$ is the Euclidean distance between node i and j, $\mathbf{e}_{ij} \in \mathbb{R}^{m_1}$ and $\mathbf{m}_{ij} \in \mathbb{R}^{m_2}$. In three-dimensional space, m_1 = 7. The features $\frac{1}{r_{i,j}}$ and $\frac{\mathbf{q}_i - \mathbf{q}_j}{r_{ij}^2}$ are motivated from DeePMD [13, 14] and $\mathbf{p}_i - \mathbf{p}_j$ brings the information of momenta. Relative coordinates and momenta are used to preserve translation invariance. The edge operator ϕ_e is an MLP. Then, edge features and messages are gathered to nodes. For node i with neighbors $\{j_k\}_{k=0}^n$, the local embedding $\mathcal{L}_i$ is computed as

$$\mathbf{E}_i = [\mathbf{e}_{i,j_0}, ..., \mathbf{e}_{i,j_{n-1}}],\ \ \mathbf{M}_i = [\mathbf{m}_{i,j_0}, ..., \mathbf{m}_{i,j_{n-1}}], \mathcal{L}_i = \mathbf{M}_i\mathbf{E}_i^T,\ \ \mathcal{L}_i \in \mathbb{R}^{m_2 \times m_1}. \tag{7}$$

Here permutation invariance is preserved by the multiplication between $\mathbf{E}_i$ and $\mathbf{M}_i$. Finally, the variation of coordinates and momenta are predicted by aggregating the contribution of each edge,

$$\Delta\mathbf{q}_{ij}, \Delta\mathbf{p}_{ij} = \phi_o(\mathcal{L}_i, \mathbf{e}_{ij}), \tag{8}$$

$$\Delta\mathbf{p}_i = \sum_{j\in\mathcal{N}(i)} \Delta\mathbf{p}_{ij},\ \ \Delta\mathbf{q}_i = \sum_{j\in\mathcal{N}(i)} \Delta\mathbf{q}_{ij} + \frac{\mathbf{p}_i}{m}\Delta t. \tag{9}$$

To ensure the conservation of momentum, for any edge, we randomly choose one direction, e.g. i -> j, compute $\Delta\mathbf{p}_{ij}$ and let $\Delta\mathbf{p}_{ji} = -\Delta\mathbf{p}_{ji}$. This treatment is also applied to the calculation of $\Delta\mathbf{q}_{ij}$'s.

4 Experiments

We test the performance of MDNet on a liquid argon system containing 4000 atoms. The interatomic interaction is modeled by Lennard-Jones potential [22]:

$$V(r) = 4\epsilon\left[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^6\right] \tag{10}$$

where r is the distance between two atoms, ϵ and σ are set to 0.2374 kcal/mol and 3.432 Å respectively. Although there is only one kind of interaction in the system, the dynamics could be rather complicated, with increasing space and time scale.

Dataset: To create the dataset, we run classical MD simulations with LAMMPS using Verlet integrator with a time step of 1 fs. We independently sample 20 trajectories from different initialized configurations, where 15 are used for training, 3 are used for testing and 2 are used for validation. For each trajectory, we randomly initialize 4000 atoms in a simulation box with periodic boundary condition in all directions. The system is first relaxed under 80 K and 1 bar in the NPT ensemble with Nosé-Hoover (NH) thermostat [23] and barostat for 20000 steps. Then, the system is further relaxed under 80 K in the NVT ensemble with NH thermostat for 20000 steps. Finally, a NVE simulation is performed for 2560 steps and only the trajectory of this final stage is used. In each trajectory, we sample 10 frames $\{\mathbf{q}(t_i), \mathbf{p}(t_i)\}_{i=1}^{10}$ of the system and use $\{\mathbf{q}(t_i + \Delta t), \mathbf{p}(t_i + \Delta t)\}_{i=1}^{10}$ as targets. According to the time reversibility of MD, the reversed data are also added to the dataset. Besides, an extra 40960-steps MD simulation in the NVT ensemble is performed to validate roll-out performance.

Implementation details: The cutoff radius r_c for constructing the nearest neighbor graph is 7.0 Å. The MLPs ϕ_e and ϕ_o both have 4 layers and use ReLU [24] as activation function. ϕ_e has 40 neurons in each hidden layer and ϕ_o has 200 neurons in each hidden layer. Training is carried out for 1000 epoch, batch size 1 with Adam optimizer [25]. Learning rate starts from 5×10^{-3} and decays exponentially at a rate of 0.8. The loss function is the mean squared error (MSE) of coordinates and momenta, normalized by their standard deviations respectively. We compare MDNet to EGNN

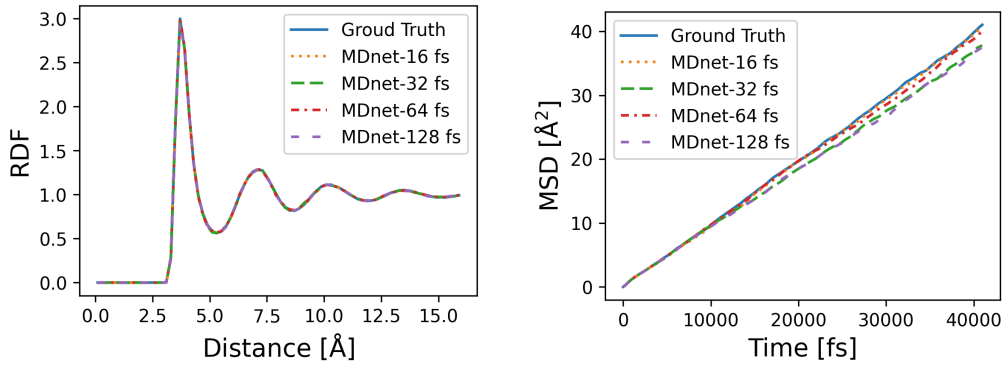

Figure 2: RDFs and MSDs computed from ground truth and MDNet with different Δt.

[20] composed of 4 EGCL layers with 64 neurons per layer and Swish activation function [26]. Hyperparameters of EGNN are consist with the original paper.

One-step results: We present the one-step root mean squared errors (RMSE) obtained by the Verlet integrator, EGNN and MDNet on validation set in Table 1. It is shown that MDNet reports comparable RMSE with Verlet with a small time step (16 fs), and significantly outperforms Verlet with a large time step. As the time step increases, the RMSE of Verlet increases faster than MDNet, due to the contribution of the fourth order error. The performance of EGNN is poor with all the time steps.

Δt [fs]	Verlet	EGNN	MDNet
16	$8.44 \times 10^{-5}, \mathbf{8.42 \times 10^{-7}}$	$2.51 \times 10^{-4}, 3.13 \times 10^{-5}$	$\mathbf{1.80 \times 10^{-5}}, 2.32 \times 10^{-6}$
32	$6.68 \times 10^{-4}, 6.58 \times 10^{-6}$	$9.94 \times 10^{-4}, 6.14 \times 10^{-5}$	$\mathbf{5.90 \times 10^{-5}}, \mathbf{4.04 \times 10^{-6}}$
64	$5.24 \times 10^{-3}, 5.74 \times 10^{-5}$	$3.82 \times 10^{-3}, 1.14 \times 10^{-4}$	$\mathbf{3.05 \times 10^{-4}}, \mathbf{1.22 \times 10^{-5}}$
128	$3.88 \times 10^{-2}, 1.75 \times 10^{-3}$	$1.34 \times 10^{-2}, 1.86 \times 10^{-4}$	$\mathbf{2.29 \times 10^{-3}}, \mathbf{5.66 \times 10^{-5}}$

Table 1: Validation RMSE for prediction of coordinates in Å (left) and velocity and Å/fs (right), respectively. Results obtained from Verlet, EGNN and MDNet with different Δt.

Roll-out performance: MD simulations can be performed by adopting one-step prediction iteratively, which is known as roll-out. We use two metrics to evaluate the roll-out performance of MDNet: radial distribution function (RDF) [27] and mean-squared displacement (MSD) [28, 29]. where RDFs provide good descriptions of equilibrium structure and MSD is a measure of transport properties of simulation systems. To compute RDFs, we conduct 40000 fs simulation and take 8 frames from the subsequent trajectory with a gap of 128 fs. The final RDFs are averaged over these 8 frames. As shown in Figure 2 (left), RDFs obtained from MDNet with different time steps coincide with ground truth, which show correct equilibrium structures. To compute MSD, the trajectory is saved with a resolution of 256 fs and calculate displacement against the first frame. As shown in Figure 2 (right), MSDs obtained from MDNet are linear with time and are in good agreement with ground truth, despite that the slopes with large time steps are slightly smaller. In summary, MDNet can reproduce the equilibrium structures and transport properties of the simulation system with large time steps. We also compare the time cost of roll-out and show that MDNet enjoys faster roll-out speed compared to Verlet, thanks to the large time step. The comparison of time cost is in the appendix B.

5 Conclusion and Discussion

We introduce MDNet, a GNN model that can predict the dynamics of many body system with large time steps. We demonstrate that MDNet exhibites good precision on one-step prediction with a large time step, and is able to generate MD trajectories. The generated trajectories are characterized by RDF and MSD, which showed good agreement with the ground truth. It is worth mentioning that MDNet only takes local environments as feature, so it could be generalized to larger systems with the same species of atoms without further training and we plan to investigate this in our future work.

There are several future directions. Firstly, the total energy is not guaranteed to be conserved during roll-out of MDNet. Incorporating thermostat into MDNet is necessary to control the temperature and total energy. Secondly, The Lennard-Jones system presented in this paper is the simplest case in MD applications. With the introduction of chemical bonds, the system will exhibit multiple characteristic times and the dynamics will be much more difficult to predict. The extension of MDNet to complicated systems is worthy of further study.

References

[1] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[2] Diddo Diddens and Andreas Heuer. Lithium ion transport mechanism in ternary polymer electrolyte-ionic liquid mixtures: A molecular dynamics simulation study. *ACS Macro Letters*, 2(4):322–326, 2013.

[3] Fatemeh Jabbari, Ali Rajabpour, and Seifollah Saedodin. Thermal conductivity and viscosity of nanofluids: a review of recent molecular dynamics studies. *Chemical Engineering Science*, 174: 67–81, 2017.

[4] Aditya K Padhi, Soumya Lipsa Rath, and Timir Tripathi. Accelerating covid-19 research using molecular dynamics simulation. *The Journal of Physical Chemistry B*, 2021.

[5] Xuewei Liu, Danfeng Shi, Shuangyan Zhou, Hongli Liu, Huanxiang Liu, and Xiaojun Yao. Molecular dynamics simulations and novel drug discovery. *Expert opinion on drug discovery*, 13(1):23–37, 2018.

[6] Loup Verlet. Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967.

[7] Sun-Ting Tsai, En-Jui Kuo, and Pratyush Tiwary. Learning molecular dynamics with simple language model built upon long short-term memory neural network. *Nature communications*, 11(1):1–11, 2020.

[8] J Kadupitiya, Geoffrey C Fox, and Vikram Jadhao. Simulating molecular dynamics with large timesteps using recurrent neural networks. *arXiv preprint arXiv:2004.06493*, 2020.

[9] Chitrak Gupta, John Kevin Cava, Daipayan Sarkar, Eric A Wilson, John Vant, Steven Murray, Abhishek Singharoy, and Shubhra Kanti Karmaker. Mind reading of the proteins: Deep-learning to forecast molecular dynamics. *bioRxiv*, 2020.

[10] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32:15379–15389, 2019.

[11] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *arXiv preprint arXiv:1909.13334*, 2019.

[12] Renyi Chen and Molei Tao. Data-driven prediction of general hamiltonian dynamics via learning exactly-symplectic maps. *arXiv preprint arXiv:2103.05632*, 2021.

[13] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14):143001, 2018.

[14] Linfeng Zhang, Jiequn Han, Han Wang, Wissam A Saidi, Roberto Car, et al. End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems. *arXiv preprint arXiv:1805.09003*, 2018.

[15] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[16] Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789*, 2019.

[17] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2019.

[18] Kailiang Wu, Tong Qin, and Dongbin Xiu. Structure-preserving method for reconstructing unknown hamiltonian systems from trajectory data. *SIAM Journal on Scientific Computing*, 42 (6):A3704–A3729, 2020.

[19] Julia Westermayr, Michael Gastegger, Maximilian FSJ Menger, Sebastian Mai, Leticia González, and Philipp Marquetand. Machine learning enables long time scale molecular photodynamics simulations. *Chemical science*, 10(35):8100–8107, 2019.

[20] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

[21] William C Swope, Hans C Andersen, Peter H Berens, and Kent R Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of chemical physics*, 76 (1):637–649, 1982.

[22] John Edward Jones. On the determination of molecular fields.—i. from the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738):441–462, 1924.

[23] William G Hoover and Brad Lee Holian. Kinetic moments method for the canonical ensemble distribution. *Physics Letters A*, 211(5):253–257, 1996.

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[27] Benjamin G Levine, John E Stone, and Axel Kohlmeyer. Fast analysis of molecular dynamics trajectories with graphics processing units—radial distribution function histogramming. *Journal of computational physics*, 230(9):3556–3569, 2011.

[28] Azam Afandak and Hossein Eslami. Ion-pairing and electrical conductivity in the ionic liquid 1-n-butyl-3-methylimidazolium methylsulfate [bmim][meso4]: molecular dynamics simulation study. *The Journal of Physical Chemistry B*, 121(32):7699–7708, 2017.

[29] Chuhong Wang, Koutarou Aoyagi, Pandu Wisesa, and Tim Mueller. Lithium ion conduction in cathode coating materials from on-the-fly machine learning. *Chemistry of Materials*, 32(9): 3741–3752, 2020.

[30] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[31] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

A The Velocity Verlet Algorithm

The algorithm of the most commonly used integrator, velocity Verlet, is:

$$\begin{aligned}
\mathbf{p}_i(t+\frac{1}{2}\Delta t) &= \mathbf{p}_i(t) + \frac{1}{2}\mathbf{F}_i(t)\Delta t, \\
\mathbf{q}_i(t+\Delta t) &= \mathbf{q}_i(t) + \frac{\mathbf{p}_i(t+\frac{1}{2}\Delta t)}{m_i}\Delta t, \\
\mathbf{p}_i(t+\Delta t) &= \mathbf{p}_i(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{F}_i(t+\Delta t)\Delta t.
\end{aligned} \tag{11}$$

In practice, $\mathbf{p}_i(t+\Delta t)$ may not need to be calculated explicitly. Then, the coordinates and momenta are updated alternately, which is also known as leapfrog integration.

B Comparison of Time Cost

We demonstrate the time cost of roll-out of MDNet and Verlet algorithm here.

	forward time [s]	time cost per fs [s]
Verlet-1 fs	3.82×10^{-3}	3.82×10^{-3}
MDNet-16 fs	4.01×10^{-2}	2.51×10^{-3}
MDNet-128 fs	4.03×10^{-2}	3.15×10^{-4}

Table 2: The time cost of Verlet and MDNet for one-step prediction and 1 fs simulation. For fair comparision, Verlet algorithm is implemented by Tensorflow [30] and MDnet by DGL [31] with Tensorflow backend. The computation time is evluated on a single Tesla P4.