Learning molecular dynamics

predicting the dynamics of glasses by a machine learning simulator

Liu, Han; Huang, Zijie; Schoenholz, Samuel S.; Cubuk, Ekin D.; Smedskjaer, Morten M.; Sun, Yizhou; Wang, Wei; Bauchy, Mathieu

Published in:
Materials Horizons

DOI (link to publication from Publisher):
10.1039/d3mh00028a

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

Watching to Simulate Glass Dynamics from Their Static Structure by Machine Learning

Han Liu [a *], Zijie Huang [b], Samuel S. Schoenholz [c], Ekin D. Cubuk [c], Morten M. Smedskjaer [d], Yizhou Sun [b], Wei Wang [b], Mathieu Bauchy [e *]

[a] *SOlids inFormaTics AI-Laboratory (SOFT-AI-Lab), College of Polymer Science and Engineering, Sichuan University, Chengdu 610065, China*

[b] *Department of Computer Science, University of California, Los Angeles, California, 90095, USA*

[c] *Brain Team, Google Research, Mountain View, California, 94043, USA*

[d] *Department of Chemistry and Bioscience, Aalborg University, Aalborg 9220, Denmark*

[e] *Physics of AmoRphous and Inorganic Solids Laboratory (PARISlab), Department of Civil and Environmental Engineering, University of California, Los Angeles, California, 90095, USA*

* *Corresponding author: Han Liu (happylife@ucla.edu), Mathieu Bauchy (bauchy@ucla.edu)*

Abstract

Many-body dynamics of atoms such as glass dynamics is generally governed by complex (and sometimes unknown) physics laws. This challenges the construction of atom dynamics simulations that both (i) capture the physics laws and (ii) run with little computation cost. Here, based on graph neural network (GNN), we introduce an observation-based graph network (OGN) framework to "*bypass all physics laws*" to simulate complex glass dynamics solely from their static structure. By taking the example of molecular dynamics (MD) simulations, we successfully apply the OGN to predict atom trajectories evolving up to a few hundred timesteps and ranging over different families of complex atomistic systems, which implies that the atom dynamics is largely encoded in their static structure in disordered phases and, furthermore, allows us to explore the capacity of OGN simulations that is potentially generic to many-body dynamics. Importantly, unlike traditional numerical simulations, the OGN simulations bypass the numerical constraint of small

integration timestep by a multiplier of $\geqslant 5$ to conserve energy and momentum until hundreds of timesteps, thus leapfrogging the execution speed of MD simulations for a modest timescale.

1. Introduction

Simulating many-body dynamics of atoms (e.g., glass dynamics) is key to predict the dynamical and transport behaviors of complex atomistic systems and to access their microscopic origins thereof [1,2]. However, the physics laws (i.e., the force-fields herein) that govern atom dynamics are essentially complex and sometimes unknown [3,4], which challenges the construction of physics-driven simulations that both (i) capture the physics laws and (ii) run with little computation cost [5]. In that regard, machine learning (ML) offers an attractive opportunity to revisit these challenges facing physics-driven simulations [6,7]. Indeed, ML excels at end-to-end learning from observed data to capture complex physics and, once trained, yields accurate-yet-fast predictions [8,9]. However, unlike studying a static system, predicting dynamics of interacting systems presents a grand challenge facing traditional ML models [10–12]—which generally fail to (i) explore the vast configuration space of an interacting system [13,14], (ii) describe the relational geometry of a configuration [14,15], (iii) infer the complex interaction modes [15,16], and (iv) conserve energy and momentum [17–19].

To mitigate the issues, graph neural network (GNN) has been recently proposed as an attractive ML model for dynamics prediction [11,15]. Unlike traditional ML models requiring human-defined structural descriptors [13,14], the GNN model directly takes as inputs the static structure and passes messages between atoms, so as to (i) keep the structural information inherently relational during prorogation [15,20] and (ii) automatically identify key structural features (if any) relevant to the dynamics [11,21]. Despite its predictive power in structural dynamics—as recently revealed in a few toy models [21,20,22–26], the potentiality of GNN remains largely untapped in simulating materials or complex interacting systems (e.g., glass dynamics) [11,21,27], which echoes a long-standing debate about *whether particle dynamics is in some way encoded in* *their static structure in disordered phases* [10,11,28]. As such, it remains elusive whether GNN could watch to simulate complex atom dynamics solely from their static structure. This question is a manifestation of a more general, grand challenge of ML in "learning complex physics from pure observations" [6,29,30]. Indeed, the underlying physics laws, no matter how complex they are, are encoded into the phenomenal observations [6,31]—from which ML may decode the laws into a surrogate ML model [6,7], which, in turn, may reduce the computational expense of physics laws [32–34] (i.e., the entire formula set of atomic force-fields and Newton's laws of motion herein [35]). However, little is known about GNN's capacity to "*bypass all physics laws*" to simulate complex atom dynamics [15,20], let alone its capacity to accelerate the simulations [4,23].

Here, based on an archetypal category of GNN termed message-passing neural network (MPNN) [36,37], we introduce an observation-based graph network (OGN) framework to "*bypass all physics laws*" to simulate complex dynamics of realistic glasses for a modest timescale, as exemplified by molecular dynamics (MD) simulations ranging over different families of complex atomistic systems that exhibit distinct types of bonds [38], including (i) binary Lennard–Jones (LJ) liquid and its melt-quenched glass [39], (ii) ionocovalent silica liquid [40], (iii) covalent silicon liquid [41], and (iv) metallic $Cu_{64.5}Zr_{35.5}$ liquid [42], which unveils the predictive power of static structure in microscopic-timescale atom trajectories (e.g., $\geqslant$ 5 timesteps per prediction for LJ liquid or potentially much longer timescale using giant OGN architecture) and, iteratively, in the short-term dynamical evolution of disordered phases up to a few hundred timesteps (e.g., $\geq$ 100 timesteps for LJ liquid) and, furthermore, allows us to explore the capacity of OGN simulations that is potentially generic to complex many-body dynamics. Importantly, by predicting $\geqslant$ 5 times longer timestep per prediction, we demonstrate that the OGN engine is computationally efficient to simulate for a short timescale of a few hundred MD steps the complex systems that are otherwise computationally expensive (or forbidden), ready to accelerate and enrich traditional simulation toolkit built upon physics laws within the scope of a modest timescale.

2. Results and Discussion

2.1 Graph analogy to MD simulation solely informed by the static structure

To establish our conclusions, we first build a ground-truth MD simulator and its analogous OGN counterpart. Figure 1A and 1B shows a schematic of the MD simulator and its OGN analogy, respectively. The MD simulator adopts a routine algorithm modelling atom dynamics according to Newton's laws of motion [5,35,43]. In detail, starting from an initial configuration that regulates the atom positions $\{r_i\}$ and velocities $\{v_i\}$, the MD algorithm consists of a loop of 4 successive steps [5,35], namely, (i) computing the system's potential energy $U(\{r_i\})$ by summing up all interatomic interactions for the current atom positions $\{r_i\}$, (ii) calculating the resultant force $\{F_i\}$ experienced by each atom i via energy differentiation (i.e., $F_i = -\partial U/\partial r_i$), (iii) obtaining each atom's acceleration $\{a_i\}$ from $\{F_i\}$ as per the Newton's law of motion, that is, $a_i = F_i/m_i$, where m_i is the mass of atom i, and finally, (iv) updating the atom positions and velocities after a small, fixed timestep via numerical integration (e.g., Verlet or leapfrog algorithm [43,44]). Eventually, this four-step loop yields the position of the atom at a function of time, that is, the atom trajectory. Note that, when computing $U(\{r_i\})$, we typically adopt a neighbor-list algorithm to reduce computation cost [35,45]. Specifically, by adopting a cutoff distance r_c to prescribe the neighbor atoms of each atom i, viz., the "neighbor-list" of atom i [45] (see Fig. 1A), the neighbor-list algorithm reduces the number of times the distance between a pair of atoms is calculated, where the interaction energy between a pair of atoms is zero if their distance is larger than r_c. Overall, the MD simulator is strictly driven by this four-step loop algorithm obeying Newton's laws of motion [35], wherein the motion of each atom is essentially governed by its complex (or unknown) interactions with its neighbor-list atoms.

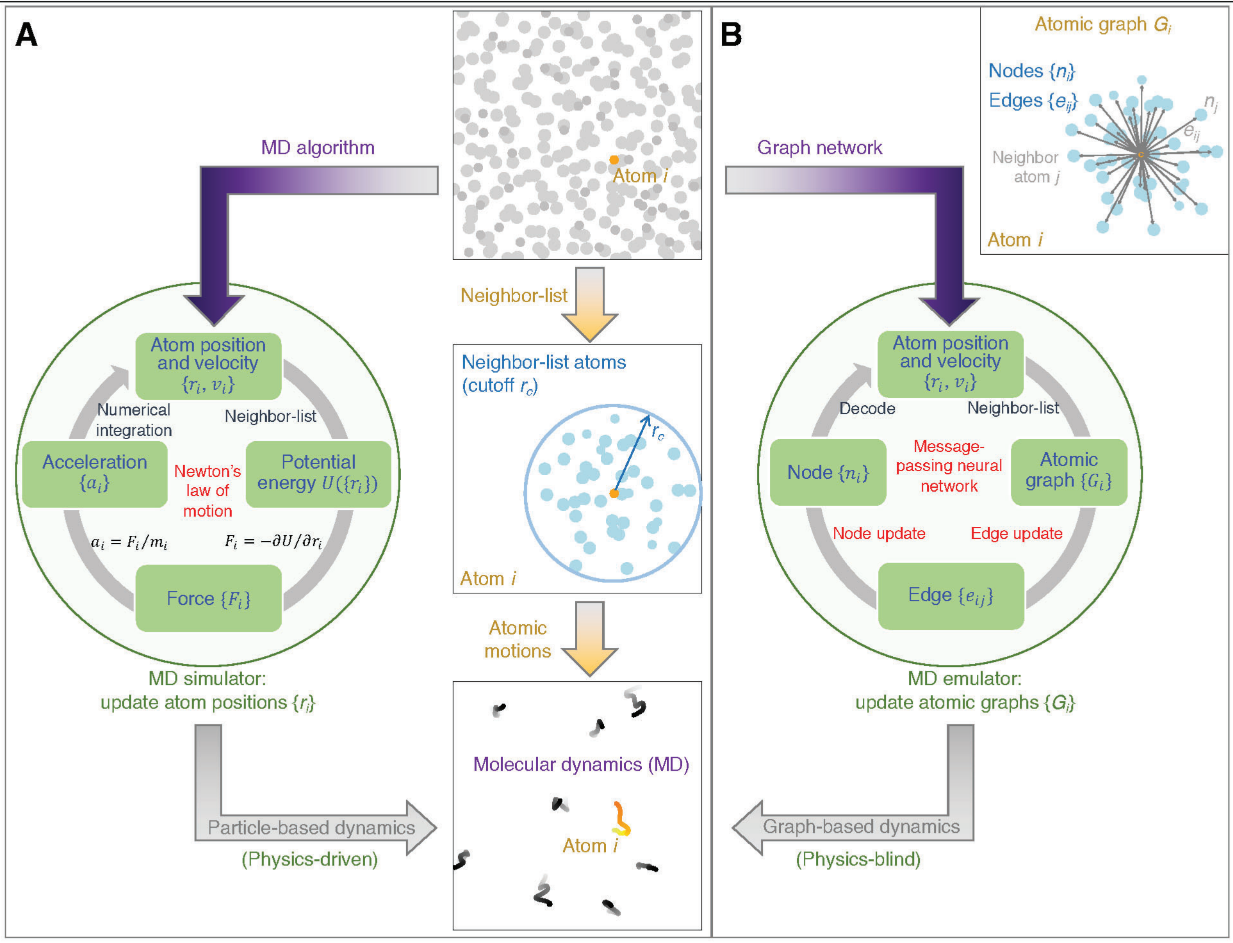

Fig. 1: Graph analogy to molecular dynamics (MD) simulation. (A) Schematic illustrating a molecular dynamics (MD) simulation that computes atomic motions by a numerical algorithm obeying Newton's law of motion [5,35] (see text for details), wherein the trajectory of each atom is governed by its interaction with its neighbor atoms within a cutoff distance (i.e., neighbor-list [35,45]). **(B)** Illustration of constructing a surrogate graph network simulation engine to predict atomic motions, wherein the neighbor-list of each atom is converted into an atomic graph built with nodes and edges representing the atoms and the interactions, respectively. Relying on message-passing neural network (MPNN) [36,37]—i.e., a graph network that takes as inputs the atomic graphs and is trained by observed atomic motions (see

text for details), the model learns to update the input graphs (i.e., edge update followed by node update) to predict the graph dynamics and the atomic motions thereof.

In analogy to the MD simulator, we build herein a surrogate graph network simulation engine solely driven by the observed structural evolution (i.e., the time-dependent atom positions and velocities) to replace the entire four-step loop of MD algorithm, as illustrated in Fig. 1B, so termed observation-based graph network (OGN). Similar to the MD simulator, the OGN is comparably driven by a four-step computation loop to predict atom dynamics, including (i) converting the neighbor-list of each atom i into an atomic graph G_i with nodes $\{n_i\}$ and edges $\{e_{ij}\}$ representing the atoms and their interactions, respectively (see Fig. 1B), (ii) updating the edges $\{e_{ij}\}$, (iii) subsequently updating the nodes $\{n_i\}$, and, finally, (iv) decoding the nodes $\{n_i\}$ to update the atom positions and velocities. Figure 2A shows the architecture of OGN built to watch atom dances and to simulate glass dynamics, where the OGN simulation engine yields the next-step configuration through 4 consecutive component layers [11,27]. Details about the four-component OGN architecture are provided in the Methods section. Notably, the OGN entirely bypasses the MD algorithm that follows Newton's law of motion [5,35] and, consequently, offers a physics-blind, closed-loop simulation engine between the present input configuration and the next-step output configuration, allowing iteratively naive prediction of atom positions and velocities as a function of time (i.e., atom dynamics).

Unlike the MD simulator that computes particle-based dynamics, the OGN predictions purely rely on graph transformation that embeds the information of atom motions. This graph-based dynamics presents a key advantage of message-passing neural network [36,37] (MPNN) architecture adopted by the OGN (see Fig. 2A), which excels at updating graph geometry relationally through message-passing between each interconnected edge and node and, in an automatic manner, identifying the pivotal, hidden structural patterns relevant to graph dynamics [11,27]—making the OGN potentially a graph analogy to MD simulation but solely informed by the static structure, namely, bypassing all physics laws to simulate atom dynamics.

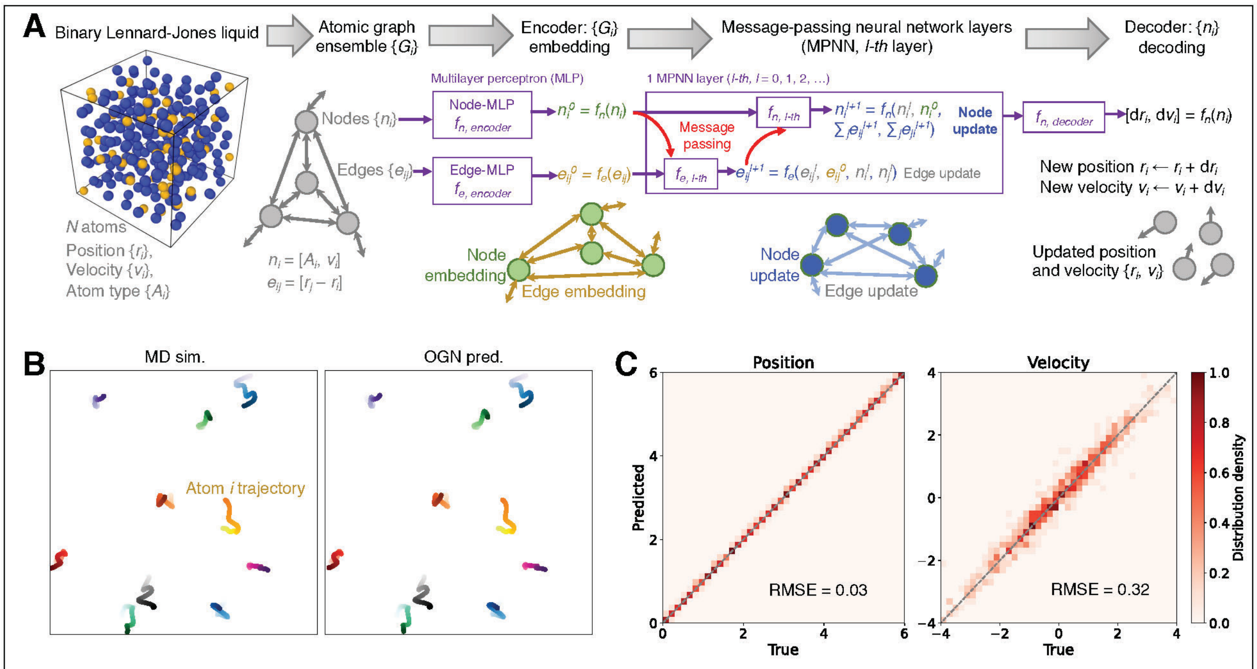
A
Binary Lennard-Jones liquid
Atomic graph ensemble {G_i}
Encoder: {G_i} embedding
Message-passing neural network layers (MPNN, l-th layer)
Decoder: {n_i} decoding
N atoms
Position {r_i}, Velocity {v_i}, Atom type {A_i}
Nodes {n_i}
Edges {e_ij}
n_i = [A_i, v_i]
e_ij = [r_j − r_i]
Multilayer perceptron (MLP)
Node-MLP f_n, encoder
Edge-MLP f_e, encoder
n_i^0 = f_n(n_i)
e_ij^0 = f_e(e_ij)
Node embedding
Edge embedding
1 MPNN layer (l-th, l = 0, 1, 2, ...)
Message passing
f_n, l-th
f_e, l-th
n_i^{l+1} = f_n(n_i^l, n_i^0, Σ_j e_ij^{l+1}, Σ_j e_ji^{l+1})
Node update
e_ij^{l+1} = f_e(e_ij^l, e_ij^0, n_i^l, n_j^l) Edge update
Node update
Edge update
f_n, decoder
[dr_i, dv_i] = f_n(n_i)
New position r_i ← r_i + dr_i
New velocity v_i ← v_i + dv_i
Updated position and velocity {r_i, v_i}
B
MD sim.
OGN pred.
Atom i trajectory
C
Position
Velocity
Predicted
True
RMSE = 0.03
RMSE = 0.32
Distribution density

Fig. 2: Observation-based graph network (OGN). (A) Schematic illustrating the architecture of observation-based graph network (OGN), which predicts the next-step change of atom positions and velocities in an input atomistic configuration, by taking the example of a binary Lennard–Jones (LJ) $A_{80}B_{20}$ liquid [39]. The OGN model consists of 4 consecutive component layers [11,27], namely, (i) the input graph layer that takes the input configuration to build atomic graphs, (ii) the encoder layer that encodes graphs, (iii) the message-passing neural network (MPNN) layers that update graphs (10 successive MPNN layers herein), and finally, (iv) the decoder layer that decodes graphs to obtain the next-step configuration (see text for details). **(B)** True (left panel) versus predicted (right panel) 100-steps atomic trajectories for randomly selected atoms in a test 265-atoms $A_{80}B_{20}$ configuration under *NVE* ensemble. LJ unit is applied. The box side length is 6.038, the neighbor-list cutoff is set as 3.0, and the timestep is set as 0.005 [39,46]. The configuration has been relaxed to an equilibrium liquid temperature $T \approx 3.0$. **(C)** Density scatter plot of the predicted versus true atom positions

(left panel) and velocities (right panel) (along x-, y-, and z-axis) in the test configuration at the last step. The $y = x$ line (grey dash) is added as a reference.

2.2 Watching to simulate Lennard–Jones system by OGN

Based on the four-component OGN framework, we now conduct an OGN simulation to predict atom dynamics from pure structural observations, by taking the example of a Kob–Andersen-type binary Lennard–Jones (LJ) $A_{80}B_{20}$ liquid, which is an archetypal model well established to investigate the generic relaxation behaviors of glassy systems governed by pairwise interactions [11,39]. Details about the MD simulation and the training process can be found in the Methods Section. In brief, we train the OGN by minimizing a loss function L that is defined as the mean square error (MSE) per atom between true versus predicted next-step output configuration, namely, $L = \sum_i(O_{i,\text{true}} - O_{i,\text{pred}})^2 / N$, wherein N is the number of atoms in the configuration, and O_i is the output next-step change of atom position dr_i and velocity dv_i for each atom i, i.e., $O_i = [dr_i, dv_i]$. Indeed, we find that the loss function L quickly reduces to a miniscule level (10^{-4}) and reaches a plateau in 1000 training epochs (see Sec. S1 in Supplementary Materials), which suggests that the OGN exhibits a powerful learning capacity of the observed atom dynamics and is able to offer an accurate prediction of next-step atomic motions.

We then use the well-trained OGN to simulate atomic motions over time (or steps) in a test configuration, as compared to the ground-truth MD simulation (see Movie S1 in Supplementary Materials). Figure 2B shows the predicted versus true 100-steps atomic trajectories of randomly selected atoms in the test configuration. Notably, the predicted trajectories exhibit an excellent agreement with that computed by the ground-truth simulation. Further, Figure 2C shows the density scatter plot of the predicted versus true atom positions and velocities (along x-, y-, and z-axis) in the test configuration at the last step. We find that both the position and velocity data points are well located in the vicinity of $y = x$ identity line. The root mean square error (RMSE) of position per atom is computed as 0.03, significantly smaller than the length scale of cage effect [47] in the LJ system (~0.4, see Sec. S2 in Supplementary Materials), which suggests that OGN simulations offer accurate predictions of not only the long-range atom migrations between vacancies [48] but also the short-range atom vibrations within a vacancy known as cage effect [47,48]. Similarly, the RMSE of velocity per atom is calculated as 0.32, an order of magnitude smaller than the velocity scale of the LJ system (i.e., $\sqrt{3.0}$ herein) [39]. Note that the velocity scale of an atomistic system is defined herein as the standard deviation of atom velocities, considering the fact that the distribution of atom velocities is approximately a gaussian distribution with a zero mean and a standard deviation of $\sqrt{(k_B T/m)}$ along *x*-, *y*-, and *z*-axis [49], where k_B is the Boltzmann constant, T is the system temperature, and m is the average atom mass. Note that, since the error will accumulate over prediction steps and lead to spurious effect in long-term dynamics [21,26,43] (up to a few hundred timesteps herein, see Sec. S3 in Supplementary Materials), we restrict herein the scope of OGN to predict the near-future atomic trajectories. Although the error accumulation surges at particle level in hundreds of timesteps, we nevertheless find that the OGN model exhibits some extent of error tolerance up to thousands of timesteps for certain system-level quantities, such as mean square displacement (MSD) and system energy (see Sec. S4 in Supplementary Materials). Overall, these results demonstrate that, without any *prior* physics knowledges, OGN can learn complex atom dynamics from pure observations of structural evolution and enables accurate predictions of near-future atomic trajectories in the LJ system.

2.3 Watching to simulate realistic glass dynamics by OGN

In analogy to OGN simulations of LJ systems, we now investigate whether the learning capacity of OGN can be generalized to atom dynamics governed by more complex interatomic interactions. In that regard, we conduct MD simulations ranging over different families of realistic atomistic systems that exhibit distinct types of bonds, as illustrated in Fig. 3A, including (i) ionocovalent silica (SiO_2) liquid governed by radial 2-body interactions comprising both the short-range pairwise interactions and the long-range

Coulombic interactions [40], (ii) covalent silicon (Si) liquid governed by not only radial 2-body interactions but also angular 3-body interactions [41], and (iii) metallic $Cu_{64.5}Zr_{35.5}$ liquid governed by many-body interactions that are decomposed into the pairwise nuclei interactions and the embedded nuclei–electron cloud interactions [42]. Details about the MD simulations and the training procedures can be found in the Methods Section. Note that we train the OGN for each of these systems in the same way as that for the LJ system, which allows us to explore the capacity of OGN simulations that is potentially generic to complex many-body dynamics.

Using the observations of these complex atom dynamics, we now examine the learning capacity of OGN to simulate these systems. Similar to LJ system, the loss function L quickly reduces to a miniscule level during training (see Sec. S1 in Supplementary Materials), so that the OGN offers an accurate prediction of next-step atomic motions for each of these complex systems. We then use the well-trained OGN to predict atomic motions in these complex systems as a function of time (see Movies S2, S3, and S4 in Supplementary Materials for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively). Figure 2B provides the true versus predicted 100-step atomic trajectories for randomly selected atoms in a test configuration for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively, wherein the settings of MD simulations and OGN architectures remains the same as that for the LJ system (see Methods Section). Notably, we find that, regardless of the nature of the interatomic interactions, OGN is able to offer an accurate prediction of near-future atomic trajectories in excellent agreement with that computed by the ground-truth simulations. Moreover, Figure 3C shows the density scatter plot of the predicted versus true atom positions and velocities (along x-, y-, and z-axis) in the test configuration at the last step for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively, wherein all the datapoints are well located in the vicinity of $y = x$ identity line to illustrate the high accuracy of OGN predictions. Further, we compute the RMSE of position and velocity for each of these systems (see Fig. 3C), which turn out to be a very miniscule error that is 1-to-2 orders of magnitude smaller than, respectively, the length scale associated with their cage effect [47,48] (see Sec. S2 in Supplementary Materials) and the velocity scale associated with their system temperature (i.e., $\sqrt{(k_B T/m)}$, see Sec. 2.2) [49], suggesting that OGN simulation is able to capture the fine details of complex atom vibration modes [48]. Overall, these results establish the conclusion that OGN is a powerful framework to simulate different systems exhibiting distinct types of bonds and is potentially generic to complex many-body dynamics. Besides that, we have also demonstrated that the OGN is a versatile tool to train efficiently by small configurations but easily generalize to simulate very large, complex systems, such as systems at different size (see Sec. S5 in Supplementary Materials), temperature (see Sec. S6 in Supplementary Materials), and density (see Sec. S7 in Supplementary Materials).

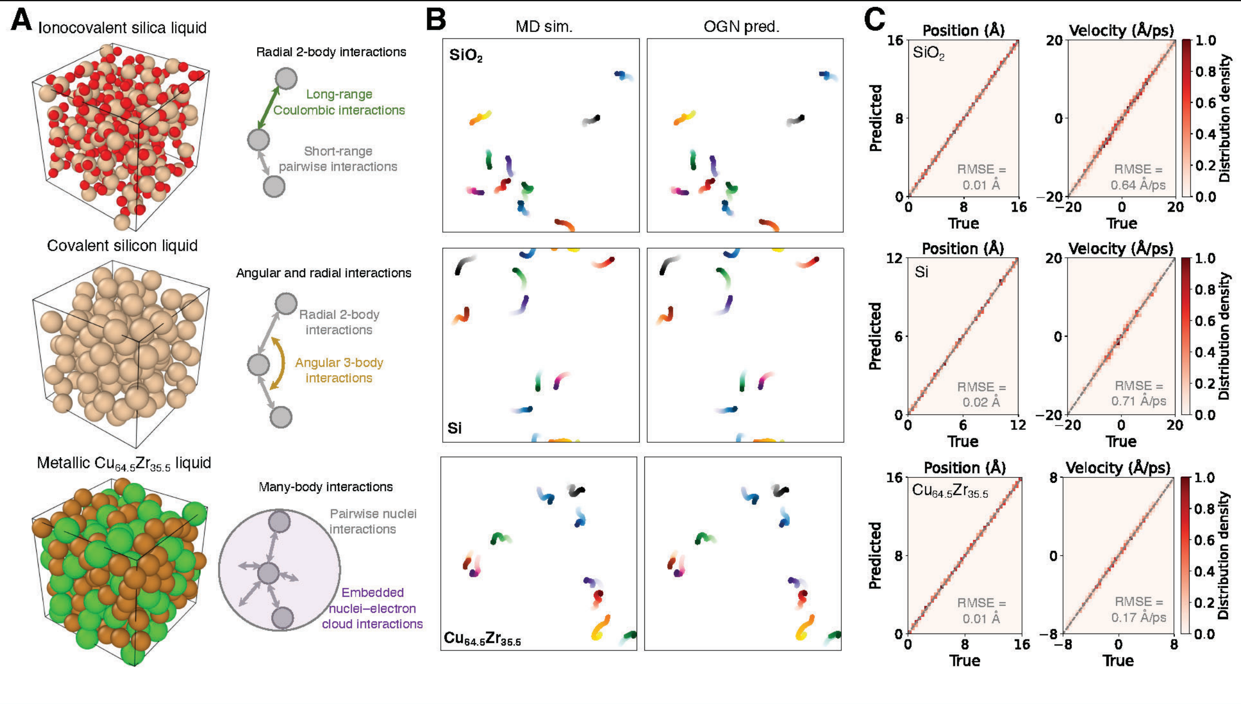

Fig. 3: Simulating complex atom dynamics by OGN. (A) Snapshots of three complex atomistic systems exhibiting distinct types of bonds, including (i) ionocovalent silica (SiO_2) liquid governed by radial 2-body interactions [40], (ii) covalent silicon (Si) liquid governed by both angular and radial interactions [41], and (iii) metallic $Cu_{64.5}Zr_{35.5}$ liquid governed by many-body interactions [42] (see text for details). The configuration built for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$ contains 363, 128, and 245 atoms, respectively, and the box side length is set to match their experimental density. **(B)** True (left panel) versus predicted (right panel) 100-steps atomic trajectories for randomly selected atoms in a test configuration under *NVE* ensemble

for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively. The liquids of SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$ have been relaxed to an equilibrium temperature around 3600 K, 2000 K, and 1500 K, respectively, and the timestep is set as 1 fs. Note that, due to its low atom diffusivity, we extend the trajectory of $Cu_{64.5}Zr_{35.5}$ to 400 steps for visibility. **(C)** Density scatter plot of the predicted versus true atom positions (left panel) and velocities (right panel) (along *x*-, *y*-, and *z*-axis) in the test configuration at the last step for SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively. The $y = x$ line (grey dash) is added as a reference.

2.4 Accelerating MD simulations by OGN

Finally, in addition to its predictive power, we investigate whether the OGN can lighten the computational burden of physics laws to accelerate MD simulations. To this end, relying on a novel automatic differentiable (auto-diff) programming platform "JAX" [50], we conduct a fair runtime comparison between OGN and the ground-truth MD simulation, i.e., the recently developed JAX-MD package [51], wherein the novel JAX platform enables computationally efficient auto-diff MD simulations [50,51]. Despite its fast execution speed [51], JAX-MD shows an intrinsic computation bottleneck arising from the small integration timestep d*t* [43], which is a strict numerical constraint rooted in numerical integration-based MD algorithms to conserve energy and momentum [43,44], as illustrated in Fig. 4A. This bottleneck presents a very general challenge facing physics-driven simulations, which are generally built upon temporospatial numerical integration [35]. In contrast, OGN is purely driven by observed data and, thus, allows us to explore the capacity of OGN simulation to bypass the small timestep d*t*, so that one OGN prediction step can span over *k* MD steps ($k > 1$) to enable the speedup of MD simulations (see Fig. 4A).

Figure 4B provides an example of the evolution of system energy and momentum with regard to MD steps for a test LJ configuration using a "Fast-OGN" by setting $k = 5$ MD steps, where (i) the kinetic, potential, and total energy and (ii) the momentum along *x*-, *y*-, and *z*-axis are computed separately to compare with their JAX-MD counterparts. Notably, despite its long timestep, the Fast-OGN remains energy and momentum conservation during a rollout of 100 MD steps, while, in contrast, JAX-MD using the same long timestep (i.e., $k = 5$ MD steps) destabilizes energy and momentum and faces some spurious effect after only a few MD steps (see Fig. 4B). Note that we restrict herein the scope of prediction to near-future atomic trajectories to avoid the spurious effect of error accumulation over iterations (see Sec. S3 in Supplementary Materials). It is worth to mention that, for each type of atomistic systems, we finely tune the Fast-OGN to best balance its prediction accuracy and execution speed (see Sec. S1 in Supplementary Materials), by (i) minimizing the number of MPNN layers until the model accuracy deteriorates severely (herein we select 2 MPNN layers, see Sec. S8 in Supplementary Materials), and, concurrently, (ii) maximizing the *k* MD steps per prediction before the input configuration loses its predictivity (herein we select $k = 5$ for LJ and 10 for other systems, see Sec. S9 in Supplementary Materials).

We now apply the Fast-OGN to make a runtime comparison with JAX-MD. Figure 4C provides the runtime comparison between JAX-MD and Fast-OGN after a rollout of 100 MD steps, as a function of system size *N* for the LJ, Si, SiO_2, and $Cu_{64.5}Zr_{35.5}$ system, respectively. As expected, the runtime cost t_c is linearly proportional to *N* (i.e., $t_c \propto N$) [45,51], where the slope represents the intrinsic runtime cost of computing all pairwise distances within a neighbor-list, and the positive intercept may arise from the inevitable computation cost of code execution in the programming platform [52,53]. We find that, except for LJ system, Fast-OGN yields a smaller slope than JAX-MD so as to enable simulation acceleration when extrapolated to large systems. Notably, when the system size increases up to $N = 10000$ atoms, it becomes evident that Fast-OGN can outperform JAX-MD with 2–10 times faster runtime for the different systems (except for the LJ system [39]—a too simple model).

Moreover, Figure 4D shows the rollout runtime of JAX-MD and Fast-OGN at $N = 10000$ atoms, as a function of the interaction complexity index—which is defined herein as the ratio of the time used to compute the empirical potential energy for a 100-atoms configuration, with respect to the time used for a reference 100-atoms LJ configuration. Obviously, since Fast-OGN is purely driven by observed atomic motions, its runtime cost is independent of the underlying complexity of interatomic interactions. In contrast, the execution speed of JAX-MD greatly relies on the computational complexity of empirical potential interactions, and from the simple LJ interaction to more complex many-body interaction (see Methods Section), finer interaction descriptions are added empirically to augment the computation burden of JAX-MD [54,55]. Overall, these results highlight the ultrafast execution speed of OGN simulations, which *bypass all physics laws*—including (i) the complexity of interatomic interactions and (ii) the numerical constraint of small integration timestep, readily accelerating interaction-complex and large-scale simulations that are otherwise computationally expensive (or forbidden).

Overall, by leveraging auto-diff programming [56], we pioneer to build, integrate, and compare physics simulator and its surrogate ML counterpart (i.e., JAX-MD [51] versus OGN) on the same platform "JAX" [50], which benefits us in several aspects. First, compared to traditional programming platforms that rely on handwritten derivatives [57], auto-diff platforms excel at computing on-the-fly the backward gradient of any quantities (e.g., force calculation in MD algorithm) with no additional computation burden associated with differentiation [50]—an operation that widely exists in ML and simulations [35,58], so as to accelerate the execution speed of ML and simulations [51]. Second, the same programming language removes communication barriers between ML and simulations, facilitating their seamless integration [5]. Third, the auto-diff JAX platform enables naive "just-in-time (JIT)" compilation of ML and simulations on high-performance hardware accelerators [50,51], and moreover, by following the same JIT rules of compilation mode and parallelization scheme [52,55], ML and simulations accelerate their code execution in the same fashion. Finally, this allows us to make a "fair" runtime comparison between OGN and JAX-MD—which is essentially a computationally-efficient reference. As such, it is remarkable that the OGN exhibits the "genuine" power to leapfrog the execution speed of MD simulations.

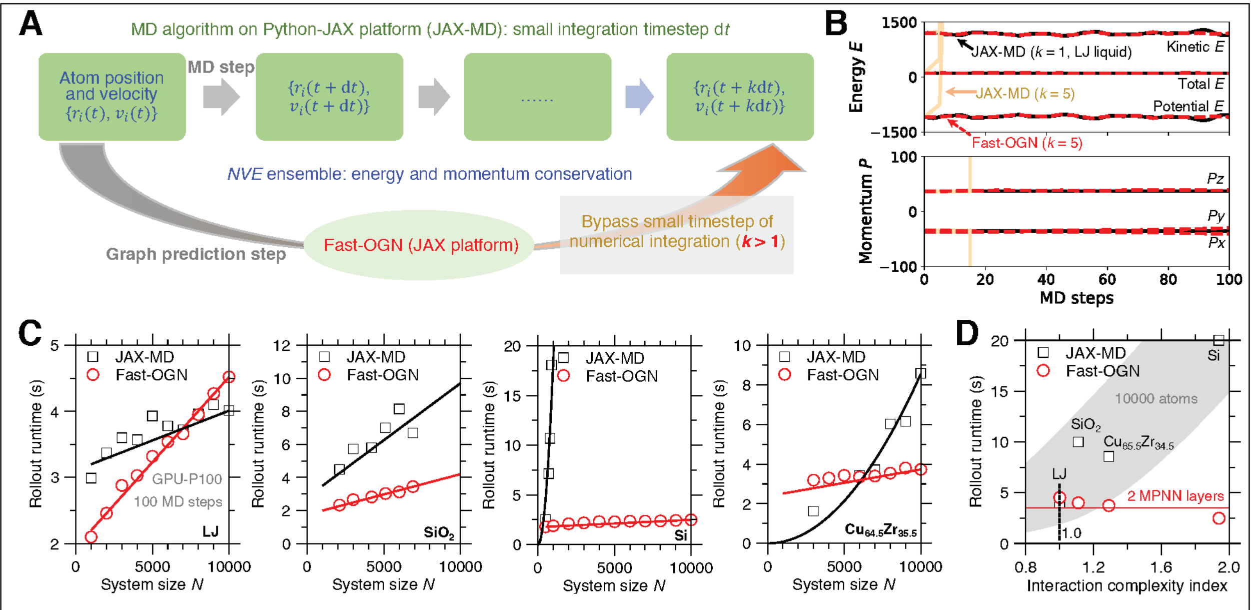

Fig. 4: Accelerating MD simulations by OGN. (A) Schematic illustrating the runtime acceleration of molecular dynamics (MD) simulation by observation-based graph network (OGN), wherein one OGN prediction step can span over k MD steps to enable the speedup of MD execution. **(B)** The evolution of kinetic, potential, and total energy (upper panel) and momentum along x-, y-, and z-axis (lower panel) with respect to MD steps for a test 265-atoms LJ configuration using a "Fast-OGN" by setting $k = 5$ MD steps per prediction (red). The JAX-MD [51] results by setting $k = 1$ (black) and $k = 5$ (orange) are added for comparison. **(C)** Runtime comparison between JAX-MD (black square) and Fast-OGN (red circle) after a rollout of 100 MD steps, as a function of system size N for LJ, SiO_2, Si, and $Cu_{64.5}Zr_{35.5}$, respectively. All computations are performed on Nvidia GPU P100 using float32 data format in Google Colab environment [53]. The lines are guides for the eyes. Note that, due to the absence of certain neighbor-list packages in the state-of-the-art version of JAX-MD [51], the computation cost of JAX-MD for Si and $Cu_{64.5}Zr_{35.5}$ shows a quadratic scaling with respect to N [51,55] (rather than a linear scaling [45]). **(D)** The rollout runtime of JAX-MD (square) and

Fast-OGN (circle) at $N = 10000$ atoms, as a function of the interaction complexity index—which is defined herein as the ratio of the computational expense between the empirical force-field and the LJ force-field (see text for details). The grey area denotes where the JAX-MD rollout runtimes are distributed. The horizonal red line is a guide of OGN rollout runtime for the eyes.

2.5 Unveiling the predictive power of liquid- versus glassy-state static structure by OGN

Finally, in addition to glass melt simulations, we apply Fast-OGN to predict atom dynamics in melt-quenched glasses featuring significantly more confined motions. Figure 5A shows the root mean square displacement as a function of time in LJ liquid and its melt-quenched glass, respectively. As expected, the glassy-state atom displacements become orders of magnitude lower than that under liquid state, which suggests that, unlike the fast relaxation of liquid, glassy-state static structure exhibits greatly delayed memory loss so that the present configuration is likely to have a stronger correlation to the next-step prediction. To this end, we train and compare two Fast-OGN models that predict liquid- and glassy-state dynamics, respectively, by taking the example of LJ liquid and its melt-quenched glass. Figure 5B provides the two models' training curves by setting the Fast-OGN timestep $k = 20$ MD steps per prediction. Indeed, we find that the Fast-OGN for glassy-state dynamics exhibits 1 order of magnitude lower prediction loss L than that for liquid-state dynamics for both the training and test sets. This confirms that, ascribed to its delayed memory loss, the glassy-state static structure exhibits more predictive power in atom dynamics for a fixed timestep than its liquid-state counterpart.

Moreover, we evaluate the one-step predictivity limit of static structure for both the LJ liquid and glass, by training Fast-OGN models over a wide range of timestep (see Sec. S9 and S10 in Supplementary Materials). Figure 5C shows the prediction loss L of, respectively, liquid- and glassy-state static structure in the test set as a function of Fast-OGN timestep. Compared to liquid-state dynamics model, we find that the glassy-state model can predict roughly 2x longer timestep per prediction, that is, from k = 5–10 MD steps (under liquid state) to k = 10–20 MD steps (under glassy state) per prediction, well before their prediction error increases exponentially with longer timestep and becomes evidently unsatisfactory. Note that, as the one-step prediction error would accumulate over iterations (see Sec. S3 in Supplementary Materials), the Fast-OGN is restricted to predict short-term atom trajectories, that is, up to ~100 MD steps for LJ liquid and ~200 MD steps for LJ glass. It is worth pointing out that the timescale reached by the iterative OGN prediction fully depends on the magnitude of one-step prediction error—which can be reduced by (i) increasing the model complexity such as the number of message-passing layers and (ii) simplifying the functional mapping such as incorporating larger neighbor list relevant to the central atom's motion during the prediction step. These model settings have been optimized to minimize the one-step prediction error (see Methods section), and we expect more endeavor in that direction to extend the prediction timescale. Overall, these results demonstrate the enhanced predictive power of static structure in glassy-state atom trajectories up to tens of MD steps per prediction and iteratively up to hundreds of MD steps, roughly 2 times longer timestep and timescale of the liquid-state atom trajectories.

It is worth mentioning that, since OGN is essentially a math operation to transform graph pattern, the theoretical implication of OGN is not simply to replace Newton's equations, but to infer the pivotal structural patterns that govern atom dynamics [11,59]. Those hidden patterns synthesized in OGN *a posterior* validate that the atom dynamics is largely encoded in their static structure, which echoes the recent finding that the topography of local energy landscape is largely encoded in the static structure [10,28]. Then the next question is: What timescale of atom dynamics can be reached by the predictive power of their static structure? Ideally, this reachable timescale refers to all timescales associated with atom reorganization in this local energy landscape of the static structure, that is, a wide spectrum of relaxation time between liquid- and glassy-state atom dynamics [11]. However, without using a giant model architecture (e.g., hundreds of deep and wide MPNN layers), the present OGN is still far from fully harnessing the predictive power of static structure. If the computational resource is unlimited, a giant-OGN architecture with considerably deep and wide MPNN layers would transform the initial graph in a very flexible and serialized manner, theoretically able to emulate much longer dynamics in one prediction step. We expect more endeavor in that direction to extend the prediction timescale. Overall, it is remarkable that, regardless of physics laws, the OGN simulation can predict near- (and potentially far-) future dynamics in one prediction step using solely the information of initial static structure—which makes OGN fundamentally different from physics-driven toolkits using infinitesimal timestep and presents a new paradigm of dynamics modeling.

Note, however, that the present shallow OGN architecture is designed to balance model accuracy and execution speed, so that the one-step predictivity become limited to restrict OGN to short-term dynamics applications. In that regard, by sacrificing execution speed, a giant-OGN architecture with deep layers of graph transformation theoretically holds the promise to predict much longer timestep per prediction step and extend to longer-term dynamics. Taking the present OGN as a basis, it remains a largely unexplored opportunity that more advanced, sophisticated OGN architecture can be developed to build a machine learning simulation engine that can extend to the targeted longer-term dynamics with a reasonable computational cost, such as the coupling of the shallow OGN module with a deep OGN module aiming to denoise the particle-level error accumulation. Although it seems unlikely to fully eliminate the propagation of errors, a deliciated design of OGN architecture and machine learning strategy (e.g., reinforcement learning to train multiple particle-level agents that can denoise particle-level errors) is likely to extend OGN to the targeted longer-term dynamics. We expect that the present work would modestly stimulate new development in that direction. Moreover, despite the requirements of long timescale in most dynamics studies, the practical applications of OGN in short-term dynamics can still intrigue some impactful outcomes. For instance, when integrating with some interpretable machine learning techniques [60,61], the OGN model is likely to offer some insights into the physics laws that governs atom dynamics—which is generally independent of timescale, such as developing an empirical forcefield from the numerous atomic trajectories in short timescale.

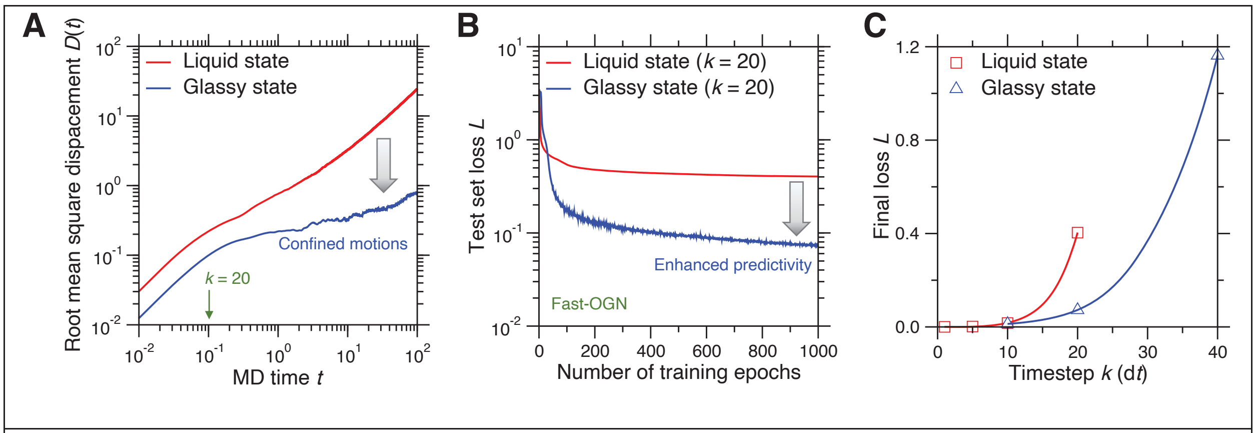

Fig. 5: One-step predictivity of Fast-OGN using liquid- versus glassy-state static structure. (A) Comparison of root mean square displacement between liquid- and glassy-state dynamics as a function of MD time, by taking the example of binary Lennard–Jones (LJ) $A_{80}B_{20}$ liquid and its melt-quenched glass [39]. **(B)** Test set loss L as a function of the number of training epochs for liquid- and glassy-state dynamics, respectively. The Fast-OGN timestep (denoted as k (dt) herein) is set as $k = 20$ MD steps per prediction. **(C)** Final loss L with respect to the Fast-OGN timestep k (dt) for liquid- and glassy-state dynamics, respectively. The lines are guides for the eyes.

3. Conclusion

Together, this work establishes the OGN simulation as an efficient paradigm to simulate in short timescale the many-body systems featuring complex dynamics (and complex physics) by solely relying on the phenomenal observations, which, in turn, unveils the predictive power of static structure in dynamical evolution of disordered phases. Importantly, the "bypassed" computational burden allows OGN simulation to readily accelerate and enrich the traditional simulation toolkit built upon physics laws within the scope of a modest timescale, that is, hundreds of MD timesteps. Future directions of OGN simulation will be placed on extending its applicability to the long-term glass dynamics, distilling the underlying interpretable physics, and enhancing the model transferability across fields of many-body dynamics. Despite its limited applicability to short-term dynamics, the OGN simulation intrigues commonalities in modeling the structural relaxation of disordered phases over different material families, microscopic interactions, and scales. This new approach holds the promise to stimulate new developments in these directions of dynamics modeling and, ultimately, facilitates the design of novel noncrystalline phases with tailored dynamical and transport properties.

4. Methods

4.1 MD simulations of four atomistic systems

(i) *Binary Lennard–Jones liquid and its melt-quenched glass.* Here, we simulate the Kob–Andersen-type binary Lennard–Jones (LJ) $A_{80}B_{20}$ liquid system (see Sec. 2.2 and 2.4) and its melt-quenched glass (see Sec. 2.5) [39], that is, an archetypal model well established to investigate the generic relaxation behaviors of glassy systems governed by pairwise interactions [11,39], where the pairwise energy U_{ij} between atom i and j is described by the general LJ potential [39]:

$$U_{ij} = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \qquad \text{Eq. (4)}$$

where r_{ij} is the interatomic distance between atom i and j, ϵ_{ij} is the minimum energy between atom i and j at their equilibrium distance r_m ($\epsilon_{AA} = 1.0$, $\epsilon_{AB} = 1.5$, and $\epsilon_{BB} = 0.5$), and σ_{ij} is a constant proportional to r_m ($\sigma_{AA} = 1.0$, $\sigma_{AB} = 0.8$, and $\sigma_{BB} = 0.88$) [39]. LJ unit is applied. The potential cutoff is 2.5, and the interactions of atom pairs with a distance larger than this cutoff are negligible and are set to zero [39].

The initial configuration adopts a cubic box with periodic boundary condition, and the side length is set as $2 \times r_c$ so as to build small-size configurations to accelerate the training of graph networks [26], where r_c is the neighbor-list cutoff and is defined as the sum of the empirical potential cutoff and the neighbor-list bin size [35,45], i.e., $r_c = 2.5 + 0.5$ (bin). The number of atoms in the configuration is set to match a preset number density of atoms $\rho_0 = 1.2$ with a deduced glass transition temperature $T_g \approx 0.3$ [39,46], i.e., system size $N = 265$ atoms. The atoms are randomly placed into the cubic box without any overlap. The atom velocities along *x*-, *y*-, and *z*-axis in the initial configuration are initialized as a normal distribution with a zero mean and a standard deviation of $\surd(k_BT/m) = \surd 3.0$ to set the system temperature as $T = 3.0$ [49], where k_B is the Boltzmann constant, and m is the average atom mass. All simulations are conducted under *NVE* ensemble. The timestep is set as 0.005 to satisfy the numerical constraint of small integration timestep for energy conservation [39,43]. The initial configuration is relaxed to an equilibrium liquid temperature around 3.0 by iteratively rescaling the distribution of atom velocities to $T = 3.0$ at each timestep until convergence, that is, multiplying each velocity by $\surd(E_K/E_{K0})$ at each timestep until $E_K \approx E_{K0}$ [49], where E_K and E_{K0} are the system's current and initial average kinetic energy per atom, respectively. This equilibrium liquid is then relaxed at $T \approx 3.0$ under *NVE* ensemble for 10000 steps to obtain the atomic trajectories. Finally, the melt-quenched glass is prepared by quenching the equilibrium liquid to a low temperature $T = 0.5$ in 10000 steps under *NVT* ensemble with a fictive temperature $T_f > 0.5$ (see Sec. S11 in Supplementary Materials). The glass is then relaxed at $T \approx 0.5$ under *NVE* ensemble for 1 million steps to obtain the atomic trajectories. All simulations are conducted using the JAX-MD package [51].

(ii) *Ionocovalent silica liquid.* The interatomic interactions in ionocovalent systems consist of both the long-range pairwise Coulombic interactions and the short-range pairwise interactions [40], which can be well described by the Buckingham-form empirical potential [62,63], and the interatomic energy U_{ij} between atoms i and j is expressed as [40,62,64]:

$$U_{ij} = \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} + A_{ij}\exp\left(-\frac{r_{ij}}{\rho_{ij}}\right) - \frac{C_{ij}}{r_{ij}^6} + \frac{D_{ij}}{r_{ij}^{24}} \qquad \text{Eq. (5)}$$

where r_{ij} is the distance between each pair of atoms, q_i is the partial charge of each atom ($q_{\mathrm{O}} = -1.047$ and $q_{\mathrm{Si}} = +2.094$ for O and Si atoms, respectively [62]), ε_0 is the dielectric constant, and A_{ij}, ρ_{ij}, C_{ij}, and D_{ij} are some parameters describing the short-range interactions. The value of A_{ij}, ρ_{ij}, C_{ij}, and D_{ij} are fixed based on Ref. [62] (viz., A_{ij} = 1386.9, 17471.7 and 0.0 eV, B_{ij} = 0.362319, 0.205205 and 1.0 Å, C_{ij} = 174.8, 133.4 and 0.0 eV·Å^6, D_{ij} = 113, 29, and 3423200 eV·Å^{24} for O–O, Si–O, and Si–Si interactions, respectively). A cutoff of 8 Å is consistently used for the short-range interactions [62]. The long-range coulombic interactions are calculated by damped shifted force (dsf) model [65] with a damping parameter of 0.25 and a cutoff of 8 Å [62]. Note that, the last term in this equation is artificially added to ensure a strong repulsion at short distance, thereby preventing any atomic overlap known as "Buckingham catastrophe" [62]. The initial configuration adopts a cubic box with periodic boundary condition and the side length is set as $2 \times r_c$, where r_c = 8.0 Å + 0.8 Å (bin) and N = 363 atoms (i.e., 121 Si atoms and 242 O atoms) so as to match the experimental density of 2.2 g/cm^3 [66]. The timestep is set as 1 fs, and the equilibrium liquid temperature is set as 3600 K [62]. We then conduct simulations under *NVE* ensemble in the same way as that for the LJ system.

(iii) *Covalent silicon liquid.* The interatomic interactions in covalent systems consist of both the radial 2-body interactions ϕ_2 and the angular 3-body interactions ϕ_3 [41], and the total potential energy U of covalent silicon system can be well described by a 3-body Stillinger–Weber (SW) empirical potential [41]:

$$U(r_{ij}, r_{ik}, \theta_{ijk}) = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j\neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk}) \qquad \text{Eq. (6)}$$

where r_{ij} is the distance between each pair of atoms, and θ_{ijk} is the angle between r_{ij} and r_{ik}. The radial 2-body interactions ϕ_2 between atom i and j is expressed as [41]:

$$\phi_2(r_{ij}) = A_{ij}\epsilon_{ij}\left[B_{ij}\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{q_{ij}}\right]\exp\left(\frac{\sigma_{ij}}{r_{ij}-a_{ij}\sigma_{ij}}\right) \qquad \text{Eq. (7)}$$

where A = 7.049556277, B = 0.6022245584, p = 4, q = 0, and a = 1.8 are some fitting parameters, σ_{ij} = 2.0951 Å is a constant proportional to the equilibrium bond length, and ϵ_{ij} = 2.1683 eV is the minimum potential energy between atom i and j at equilibrium [41]. The angular 3-body interactions ϕ_3 of atom i with respect to its two neighbors j and k is expressed as [41]:

$$\phi_3(r_{ij}, r_{ik}, \theta_{ijk}) = \lambda_{ijk}\epsilon_{ijk}(\cos\theta_{ijk} - \cos\theta_{0ijk})^2 \exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij}-a_{ij}\sigma_{ij}}\right)\exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik}-a_{ik}\sigma_{ik}}\right) \qquad \text{Eq. (8)}$$

where $\gamma = 1.2$ is a fitting parameter, $\theta_{0ijk} = 109°$ is the preferred energy-stable angle between r_{ij} and r_{ik}, and $\epsilon_{ijk} = 2.1683$ eV and $\lambda_{ijk} = 21$ are the penalty energy and its coefficient, respectively [41]. The SW potential has an automatic cutoff at $a\sigma = 3.77$ Å [41], and the neighbor-list cutoff r_c is herein set as 3.77 Å + 1.05 Å (bin). The initial configuration contains 128 atoms in a cubic box with periodic boundary condition, and the side length is set as 13.45 Å, in accordance with the experimental density of 2.53 g/cm^3 [67]. The timestep is set as 1 fs, and the equilibrium liquid temperature is set as 2000 K [41]. We then conduct simulations under *NVE* ensemble in the same way as that for the LJ system.

(iv) *Metallic $Cu_{64.5}Zr_{35.5}$ liquid.* The interatomic interactions in metallic systems are many-body interactions consisting of both the pairwise nuclei interactions and the embedded nuclei–electron cloud interactions [42], which can be well described by the Embedded Atom Method (EAM) potential [68], and the potential energy U_i of a central atom i is formulated as [42]:

$$U_i = F_\alpha\left(\sum_{j\neq i} \rho_\beta\,(r_{ij})\right) + \frac{1}{2}\sum_{j\neq i} \phi_{\alpha\beta}(r_{ij}) \qquad \text{Eq. (9)}$$

where F is the energy gained by embedding the cation i in the "ocean" of delocalized electrons described by the local atomic electron density ρ, ϕ is a pair potential interaction describing the cation-cation interactions, r_{ij} is the interatomic distance between atom i and j, α, β represent element type of atom i and j, respectively, and j denotes the neighbors of atom i within a radius cutoff (7.6 Å for $Cu_{64.5}Zr_{35.5}$) [42]. The function profile of F, ρ, and ϕ for $Cu_{64.5}Zr_{35.5}$ is provided by Ref. [42]. The initial configuration adopts a cubic box with periodic boundary condition and the side length is set as $2 \times r_c$, where $r_c = 7.6$ Å + 0.4 Å (bin) and $N = 245$ atoms (i.e., 158 Cu atoms and 87 Zr atoms) so as to match the experimental density of 59.32 atom/nm^3 [42]. The timestep is set as 1 fs, and the equilibrium liquid temperature is set as 1500 K [42]. Finally, all simulations are conducted under *NVE* ensemble in the same way as that for the LJ system using the JAX-MD package [51].

4.2 OGN model architecture

We now take a closer inspection into the OGN functionality. Figure 2a shows the architecture of OGN built to watch atom dances and to simulate glass dynamics. Starting from an N-atoms input configuration with the information of atom positions $\{r_i\} \in \mathbb{R}^{\mathrm{N} \times 3}$, velocities $\{v_i\} \in \mathbb{R}^{\mathrm{N} \times 3}$, and one-hot representation [58] of atom types $\{A_i\}$ (e.g., A_i = [1, 0] or [0, 1] for, respectively, A- or B-type atom i in a binary system), the OGN simulation engine yields the next-step configuration through 4 consecutive component layers [11,27]:

(i) ***the input graph layer*** that builds atomic graphs $\{G_i\}$ by converting the neighbor-list of each atom i into a geometric graph G_i comprising nodes $\{n_i\}$ and edges $\{e_{ij}\}$, where the node representation of atom i is $n_i = [A_i, v_i]$ (i.e., the atom type and velocity) and the edge representation between atom i and j is $e_{ij} = [r_j - r_i]$ (i.e., a directional distance between the two atoms).

(ii) ***the encoder layer*** that encodes graphs, where the encoder contains a node-MLP (i.e., multilayer perceptron [58]) function $f_{n,encoder}$ and an edge-MLP function $f_{e,encoder}$ that compute, respectively, the embedding n_i^0 of each node n_i (i.e., $n_i^0 = f_{n,encoder}(n_i)$) and the embedding e_{ij}^0 of each edge e_{ij} (i.e., $e_{ij}^0 = f_{e,encoder}(e_{ij})$).

(iii) ***the successive MPNN layers*** that update graphs, where the l-th MPNN layer (l = 0, 1, 2, ...) updates the edges $\{e_{ij}^l\}$ and nodes $\{n_i^l\}$ from previous layer by a sequential operation of edge update followed by node update [11,27], namely, first using an edge-MLP function f_e^l to compute the edge update e_{ij}^{l+1}, that is,

$$e_{ij}^{l+1} = f_e^l(e_{ij}^l, n_i^l, n_j^l) \qquad \text{Eq. (10)}$$

where the information of the two end nodes n_i^l and n_j^l are passed into the edge e_{ij}^l, and then using a node-MLP function f_n^l to compute the node update n_i^{l+1}, that is,

$$n_i^{l+1} = f_n^l(n_i^l, \textstyle\sum_j e_{ij}^{l+1}, \sum_j e_{ji}^{l+1}) \qquad \text{Eq. (11)}$$

where the aggregation information of the updated edges $\sum_j e_{ij}^{l+1}$ (outgoing edges) and $\sum_j e_{ji}^{l+1}$ (incoming edges) are passed into the node n_i^l. Note that, the message passing between nodes and edges is key to keep the graph geometry inherently relational during propagation and allows the OGN to automatically identify the non-intuitive, pivotal structural patterns relevant to graph dynamics [11,27,59].

(iv) ***the decoder layer*** that decodes graphs, where the decoder is a node-MLP function $f_{n,decoder}$ that transforms the updated nodes $\{n_i\}$ into the next-step change of atom positions $\{\mathrm{d}r_i\}$ and velocities $\{\mathrm{d}v_i\}$, i.e., $[dr_i, dv_i] = f_{n,decoder}(n_i)$, so as to yield the next-step configuration. More details about the model settings are described in the following section.

4.3 OGN model settings

Based on the four-component OGN framework, we describe herein several settings key to the OGN's learning capability (i.e., the training performance), including:

(i) *MLP functional.* The node- and edge-MLP functions can exhibit different complexity of neural network representations [21], which, herein, are all set as the MLP consisting of one hidden layer (64 neurons, ReLU activation) followed by an output layer (64 neurons, ReLU activation) [21]. Note that the decoder has a non-activated output layer containing 6 neurons (i.e., outputting $\mathrm{d}r_i$ and $\mathrm{d}v_i$ along x-, y-, and z-axis).

(ii) *LayerNorm layer.* In accordance with the dataset standardization (see below), all MLP (except the decoder) are followed by a LayerNorm layer [69]—which generally improves the training stability [21]. Note that, unlike the dataset standardization that normalizes each element in a node (or edge) array n_i over all the nodes (or edges) $\{n_i\}$ in the dataset [58], the LayerNorm operation normalizes each element in the array representation of a single node (or edge) n_i over all elements in the array n_i [69], which is generally found to stabilize the training of hidden neural layers [69].

(iii) *Graph concatenation.* Further, we stabilize the training of the successive MPNN layers by concatenating the input graph features $\{n_i^l\}$ and $\{e_{ij}^l\}$ at each MPNN layer with the constant graph embeddings $\{n_i^0\}$ and $\{e_{ij}^0\}$ offered by the encoder [11], namely, $n_i^l = [n_i^l, n_i^0]$ and $e_{ij}^l = [e_{ij}^l, e_{ij}^0]$ at the l-th MPNN layer (see Fig. 2A).

(iv) *Number of MPNN layers.* Finally, by fixing all these settings above, the OGN's learning performance mainly relies on the number of MPNN layers, and more successive MPNN layers can significantly improve the model complexity and, therefore, enhance the prediction accuracy [21,26]. Moreover, more layer-by-layer message-passing allow each node to receive the updated message from further distant nodes and edges (beyond the neighbor-list cutoff r_c) that may potentially affect the dynamics of the central node [11,27]. Despite the fact that more MPNN layers yields more accurate prediction, we find that the training performance remains satisfactory even if the OGN is simplified to adopt only one MPNN layer (see Sec. S8 in Supplementary Materials), as the model has already imbibed the entire neighbor-list atoms that account for even the weakest interactions (at the distance $r \approx r_c$) responsible to the atom dynamics. When the layer number $l > 2$, we find that the enhancement of OGN's learning capacity becomes inconsiderable for our dataset (see Sec. S8 in Supplementary Materials), which is likely ascribed to the fact that the configurations in our training set are built using a small box size of $2 \times r_c$ to promote the training efficiency, so that the update message of every node and edge (when passing from the 1st to 2nd MPNN layer) has been propagated throughout the entire atomistic configuration [11,27]. Here, the OGN adopts 10 MPNN layers to offer an unlimited learning capacity (see Sec. 2.2 to 2.3). However, 2 MPNN layers nevertheless allows us to construct a Fast-OGN (see Sec. 2.4 to 2.5) that offers a satisfactory prediction accuracy (see Sec. S1 and S8 in Supplementary Materials).

4.4 OGN training procedure

All machine learning procedures are performed on the JAX programming platform [50], and we describe herein several key steps of the training procedure:

(i) *Training and test sets.* The training set is built upon 10000 pairs of the current and next-step configurations provided by 10 independent 1000-steps MD trajectories. In detail, the current N-atoms configuration is converted into N input atomic graphs $\{G_i\}$, and all N atomic graphs $\{G_i\}$ together constitute an N-atom graph batch, which consists of a complete, deduplicated set of nodes $\{n_i\}$ and edges $\{e_{ij}\}$ (bidirectional) that represent the N atoms and their interactions in the configuration (see Fig. 2A), respectively. We input this complete set of $\{n_i\}$ and $\{e_{ij}\}$ as one batch into the OGN model to calculate the loss function L (see below), where the target output of OGN, i.e., the next-step change of atom positions $\{\mathrm{d}r_i\}$ and velocities $\{\mathrm{d}v_i\}$ for the N atoms in the batch, is obtained from the next-step configuration. In other words, the training set contains 10000 batches, and each batch contains an N-atoms configuration pair, that is, the current configuration (converted to a complete set of $\{n_i\}$ and $\{e_{ij}\}$) as input and the next-step configuration (converted to the N atoms' $\{\mathrm{d}r_i\}$ and $\{\mathrm{d}v_i\}$) as output. Similarly, the test set contains 100 batches provided by 100 independent pairs of input and output configurations.

(ii) *Online standardization.* Both the input (i.e., $\{n_i\}$ and $\{e_{ij}\}$) and output (i.e., $\{\mathrm{d}r_i\}$ and $\{\mathrm{d}v_i\}$) in the training and test sets have been online standardized with respect to the past detected training set so as to accelerate the training [21,58]. Namely, every time before an input n_i, for instance, is fed into the OGN model, it is normalized by the mean and variance of all the past values of n_i seen by the model [21,58]. Dataset standardization is generally found to reduce training time [58], and we adopt herein the online standardization technique to account for the augment of training set from rotating each input training configuration [11,21] (see below).

(iii) *Loss function.* The loss function L is defined as the mean square error (MSE) per atom between the true versus predicted outputs $\{O_{i,\mathrm{true}}\}$ versus $\{O_{i,\mathrm{pred}}\}$ for an N-atoms configuration, that is, $L = \Sigma_i(O_{i,\mathrm{true}} - O_{i,\mathrm{pred}})^2 / N$, where $\{O_i\}$ is the next-step change of atom positions $\{\mathrm{d}r_i\}$ and velocities $\{\mathrm{d}v_i\}$, i.e., $O_i = [dr_i, dv_i]$, and L is the average loss over each elements in the output array O_i. Note that, since the outputs $\{\mathrm{d}r_i\}$ and $\{\mathrm{d}v_i\}$ have been standardized, the loss function L is a standardized loss accordingly [21].

(iv) *Initialization.* In accordance with the dataset standardization [58] and the use of LayerNorm layers [69] that regulate the unit magnitude of the loss function landscape [21,58], the weights and bias in each neuron are initialized from a truncated normal distribution with a mean of zero and a standard deviation of $1/n$ (herein, n is the number of weights) [11,58], which effectively tunes the magnitude of each neuron output to the unit scale and offers a reasonable initialization in the loss function landscape, so as to reduce the training time and improve the training stability [11,21,58].

(v) *Learning rate.* The learning rate (LR) is set as an exponential decay from 10^{-4} to 10^{-6} in 20 million gradient update steps [21], i.e., LR = $10^{-4} \times (0.1$ ^ $(K / 10^7))$, where K is the number of gradient update steps. In practice, each gradient update step corresponds to a batch (i.e., a pair of input and output configurations) in the training set used to compute a loss L.

(vi) *Training epochs.* Once all the settings above have been fixed, we start to minimize the loss function L as a function of the neuron network hyperparameters in the OGN model, by using the training set that contains 10000 batches (i.e., 10000 pairs of input and out configurations). We perform the training to 1000 epochs, where each epoch covers the 10000 batches in the training set, and each batch yields a loss L to adjust the OGN hyperparameters by gradient backpropagation training [58]. During training, we record the model accuracy every 1000 training batches by scanning over the 100 batches in the test set and computing the average loss L for the test set. Depending on the number of MPNN layers and the size of atomic graphs, the training typically takes a few days (~2-7 days) to finish on the JAX programming platform that is naively complied on the Nvidia GPU V100 hardware using float32 data format [50].

(vii) *Random symmetry per epoch.* Finally, it should be pointed out that, during training, the configurations in the training set are randomly subjected to one of the symmetries of a cubic box (i.e., reflection and rotation) at each training epoch to augment the training set [11]. Since the present OGN model is invariant to geometric translation—by using relative atom positions in the input atomic graphs—but not invariant to geometric reflection and rotation [23], this dataset augment allows the OGN to learn the symmetry of graph geometry [11,23].

Conflict of Interest

The authors declare no competing financial interests.

Acknowledgements

H.L. acknowledges funding from the Fundamental Research Funds for the Central Universities under the Grant No. YJ202271. M.B. acknowledges the National Science Foundation under Grants No. DMR-1928538, DMR-1944510, and DMREF-1922167. Parts of the computational work were performed on TianHe-1(A) at National Supercomputer Center in Tianjin.

Reference

1 C. Massobrio, Ed., *Molecular dynamics simulations of disordered materials: from network glasses to phase-change memory allyos*, Springer, Cham Heidelberg, 2015.

2 E. D. Cubuk, R. J. S. Ivancic, S. S. Schoenholz, D. J. Strickland, A. Basu, Z. S. Davidson, J. Fontaine, J. L. Hor, Y.-R. Huang, Y. Jiang, N. C. Keim, K. D. Koshigan, J. A. Lefever, T. Liu, X.-G. Ma, D. J. Magagnosc, E. Morrow, C. P. Ortiz, J. M. Rieser, A. Shavit, T. Still, Y. Xu, Y. Zhang, K. N. Nordstrom, P. E. Arratia, R. W. Carpick, D. J. Durian, Z. Fakhraai, D. J. Jerolmack, D. Lee, J. Li, R. Riggleman, K. T. Turner, A. G. Yodh, D. S. Gianola and A. J. Liu, *Science*, 2017, **358**, 1033–1037.

3 E. Paquet and H. L. Viktor, *BioMed Research International*, 2015, **2015**, e183918.

4 P. Friederich, F. Häse, J. Proppe and A. Aspuru-Guzik, *Nature Materials*, 2021, **20**, 750–761.

5 H. Liu, Z. Zhao, Q. Zhou, R. Chen, K. Yang, Z. Wang, L. Tang and M. Bauchy, *Comptes Rendus. Géoscience*, 2022, **354**, 1–43.

6 G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto and L. Zdeborová, *Rev. Mod. Phys.*, 2019, **91**, 045002.

7 G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, *Nature Reviews Physics*, 2021, **3**, 422–440.

8 H. Liu, Z. Fu, K. Yang, X. Xu and M. Bauchy, *Journal of Non-Crystalline Solids: X*, 2019, **4**, 100036.

9 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, *Nature*, 2018, **559**, 547.

10 H. Liu, S. Xiao, L. Tang, E. Bao, E. Li, C. Yang, Z. Zhao, G. Sant, M. M. Smedskjaer, L. Guo and M. Bauchy, *Acta Materialia*, 2021, **210**, 116817.

11 V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. R. Nelson, T. Back, D. Hassabis and P. Kohli, *Nat. Phys.*, 2020, **16**, 448–454.

12 Z. Fan, J. Ding and E. Ma, *Materials Today*, 2020, **40**, 48–62.

13 A. P. Bartók, R. Kondor and G. Csányi, *Physical Review B*, , DOI:10.1103/PhysRevB.87.184115.

14 G. P. P. Pun, R. Batra, R. Ramprasad and Y. Mishin, *Nature Communications*, 2019, **10**, 2339.

15 P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende and K. Kavukcuoglu, *arXiv:1612.00222 [cs]*.

16 K. Xu, J. Li, M. Zhang, S. S. Du, K. Kawarabayashi and S. Jegelka, *arXiv:2009.11848 [cs, stat]*.

17 S. Greydanus, M. Dzamba and J. Yosinski, in *Advances in Neural Information Processing Systems 32*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox and R. Garnett, Curran Associates, Inc., 2019, pp. 15379–15389.

18 M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel and S. Ho, *arXiv:2003.04630 [physics, stat]*.

19 Y. D. Zhong, B. Dey and A. Chakraborty, 12.

20 T. Kipf, E. Fetaya, K.-C. Wang, M. Welling and R. Zemel, in *International Conference on Machine Learning*, PMLR, 2018, pp. 2688–2697.

21 A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec and P. W. Battaglia, *arXiv:2002.09405 [physics, stat]*.

22 Z. Huang, Y. Sun and W. Wang, 11.

23 V. G. Satorras, E. Hoogeboom and M. Welling, 16.

24 J. Brandstetter, R. Hesselink, E. van der Pol, E. Bekkers and M. Welling, *arXiv:2110.02905 [cs, stat]*.

25 T. Xie, A. France-Lanord, Y. Wang, Y. Shao-Horn and J. C. Grossman, *Nat Commun*, 2019, **10**, 2667.

26 T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez and P. W. Battaglia, *arXiv:2010.03409 [cs]*.

27 P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li and R. Pascanu, *arXiv:1806.01261 [cs, stat]*.

28 H. Liu, M. M. Smedskjaer and M. Bauchy, *PHYSICAL REVIEW B*.

29 R. S. Michalski and R. E. Stepp, in *Machine Learning: An Artificial Intelligence Approach*, eds. R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Springer, Berlin, Heidelberg, 1983, pp. 331–363.

30 B. Zhu, S. Wang and J. Zhang, *arXiv:2006.05044 [cs, stat]*.

31 P. Kroupa, *Can. J. Phys.*, 2015, **93**, 169–202.

32 D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner and S. Hoyer, *arXiv:2102.01010 [physics]*.

33 M. F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. H. Froula, G. Gregori, M. Jarvis, S. Khatiwala, J. Korenaga, J. Topp-Mugglestone, E. Viezzer and S. M. Vinko, *arXiv:2001.08055 [physics, stat]*.

34 D. M. de O. Zapiain, J. A. Stewart and R. Dingreville, *npj Comput Mater*, 2021, **7**, 1–11.

35 M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, 2017.

36 J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, in *International Conference on Machine Learning*, PMLR, 2017, pp. 1263–1272.

37 J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, in *Machine Learning Meets Quantum Physics*, eds. K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda and K.-R. Müller, Springer International Publishing, Cham, 2020, pp. 199–214.

38 L. Tang, H. Liu, G. Ma, T. Du, N. Mousseau, W. Zhou and M. Bauchy, *Mater. Horiz.*, , DOI:10.1039/D0MH00980F.

39 W. Kob and H. C. Andersen, *Physical Review Letters*, 1994, **73**, 1376–1379.

40 H. Liu, Z. Fu, Y. Li, N. F. A. Sabri and M. Bauchy, *MRS Communications*, 2019, 1–7.

41 F. H. Stillinger and T. A. Weber, *Physical Review B*, 1985, **31**, 5262–5271.

42 M. I. Mendelev, M. J. Kramer, R. T. Ott, D. J. Sordelet, D. Yagodin and P. Popel, *Philosophical Magazine*, 2009, **89**, 967–987.

43 J. Du, in *Springer Handbook of Glass*, eds. J. D. Musgraves, J. Hu and L. Calvez, Springer International Publishing, Cham, 2019, pp. 1131–1155.

44 L. Verlet, *Phys. Rev.*, 1967, **159**, 98–103.

45 A. A. Chialvo and P. G. Debenedetti, *Computer Physics Communications*, 1990, **60**, 215–224.

46 L. Wang and N. Xu, *Phys. Rev. Lett.*, 2014, **112**, 055701.

47 B. Li, K. Lou, W. Kob and S. Granick, *Nature*, 2020, **587**, 225–229.

48 M. Bauchy and M. Micoulaut, *Phys. Rev. B*, 2011, **83**, 184118.

49 P. A. Tipler and G. Mosca, *Physics for Scientists and Engineers*, Macmillan, 2007.

50 J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne and Q. Zhang, JAX: composable transformations of Python+NumPy programs (version 0.2.5) http://github.com/google/jax 2018.

51 S. Schoenholz and E. D. Cubuk, *Advances in Neural Information Processing Systems*, 2020, **33**, 11428–11441.

52 M. Gecht, M. Siggel, M. Linke, G. Hummer and J. Köfinger, *The Journal*, 2020, 18.

53 E. Bisong, in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, ed. E. Bisong, Apress, Berkeley, CA, 2019, pp. 59–64.

54 J. A. Harrison, J. D. Schall, S. Maskey, P. T. Mikulski, M. T. Knippenberg and B. H. Morrow, *Applied Physics Reviews*, 2018, **5**, 031104.

55 A. Yaseen, H. Ji and Y. Li, *Journal of Parallel and Distributed Computing*, 2016, **87**, 91–101.

56G. Corliss, C. Faure, A. Griewank, L. Hascoet and U. Naumann, *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Springer Science & Business Media, 2013.

57 S. Plimpton, *Journal of Computational Physics*, 1995, **117**, 1–19.

58 E. Alpaydin, *Introduction to Machine Learning*, MIT Press, 2014.

59 E. Boattini, F. Smallenburg and L. Filion, *Phys. Rev. Lett.*, 2021, **127**, 088007.

60 M. D. Cranmer, R. Xu, P. Battaglia and S. Ho, *arXiv:1909.05862 [astro-ph, physics:physics, stat]*.

61 H. Liu, F.-Y. Wu, G.-J. Zhong and Z.-M. Li, *Materials & Design*, 2023, **227**, 111773.

62 H. Liu, Y. Li, Z. Fu, K. Li and M. Bauchy, *J. Chem. Phys.*, 2020, **152**, 051101.

63 B. W. H. van Beest, G. J. Kramer and R. A. van Santen, *Physical Review Letters*, 1990, **64**, 1955–1958.

64 H. Liu, Z. Fu, Y. Li, N. F. A. Sabri and M. Bauchy, *Journal of Non-Crystalline Solids*, 2019, **515**, 133–142.

65 C. J. Fennell and J. D. Gezelter, *The Journal of Chemical Physics*, 2006, **124**, 234104.

66 N. P. Bansal and R. H. Doremus, *Handbook of Glass Properties*, Elsevier, 2013.

67 V. M. Glazov, S. N. Chizhevskaia and N. N. Glagoleva, *Liquid semiconductors*, Plenum Press, New York, 1969.

68 M. S. Daw, S. M. Foiles and M. I. Baskes, *Materials Science Reports*, 1993, **9**, 251–310.

69 J. L. Ba, J. R. Kiros and G. E. Hinton, *arXiv:1607.06450 [cs, stat]*.