Politecnico di Torino

université PARIS-SACLAY

Master's Degree in Physics of Complex Systems

GRAPH NEURAL NETWORKS FOR GLASSY MATERIALS

Supervisors

Prof. François LANDES

Prof. Andrea GAMBA

Candidate

Francesco Saverio PEZZICOLI

October 2021

Abstract

In fundamental physics, a crucial and unsolved problem is that of understanding the behavior of glassy liquids. In these materials the viscosity or any other characteristic time increases very quickly (about 13 orders of magnitude) when the temperature is varied by only a few tens of percent around a characteristic temperature T_g, without any obvious change in the geometrical structure of their elementary constituents. This raises the question: is structure important to glassy dynamics? Several studies have shown that a lot of information about the dynamical behaviour of these materials is contained into the static structure, but how to extract this information is still an open question. A recent branch of research focuses on applying machine learning (ML) methods to extract information from static structure. Here we pursue this direction by combining a powerful family of ML models, Graph Neural Networks, with expert knowledge in the field to reach better understanding of these materials and develop new ML tools.

Acknowledgements

Table of Contents

Chapter 1

Introduction

In fundamental physics, a crucial and unsolved problem is that of understanding the behavior of glassy liquids. In these materials the viscosity or any other characteristic time increases very quickly (about 13 orders of magnitude) when the temperature is varied by only a few tens of percent around a characteristic temperature T_g, without any obvious change in the geometrical structure of their elementary constituents. This raises the question: **is structure important to glassy dynamics?** Several studies [1][2][3] have shown that a lot of information about the dynamical behaviour of these materials is contained into the static structure, but how to extract this information is still an open question. Physicists have been trying for years to characterize the so called dynamic glass transition, looking for a diverging length-scale. This requires defining a suitable order parameter and measuring its relative correlation function, but no satisfactory answer was found. Another branch of research focuses on **applying machine learning (ML) methods to extract information from static structure**. Here we pursue this direction combining a powerful family of ML models, Graph Neural Networks, with expert knowledge in the field to reach better understanding of these materials and develop new ML tools.

1.1 Supercooled liquids phenomenology

In our work we are going to focus on supercooled liquids. These are liquids that are equilibrated under the melting temperature T_m into a **metastable phase** in such a way that time-translation invariance (and thus the fluctuation dissipation theorem) holds. In short, supercooled liquids show the amorphous, **disordered structure** of classical liquids at equilibrium under the melting temperature. In the following we will characterize the phenomenology of these materials following the steps of [4] which is an excellent introductory review in the field.

To obtain supercooled liquids one must adopt a particular cooling protocol in order to avoid two processes: **crystallization** and going **out of equilibrium**. Under T_m the actual equilibrium state of the system is the crystal phase, indeed, in numerical models of glass-forming liquids, the lowest energy is reached with a regular, ordered, arrangement of particles. Thus when we cool our sample we have to **avoid the nucleation** of the stable phase (crystal) from the metastable one (liquid). At the same time we want to keep the sample at equilibrium: if the liquid is cooled too quickly it cannot equilibrate, time translation invariance is broken, two-times correlation functions do not depend simply on the difference of times anymore and the fluctuation dissipation relation is broken. When this happens we say that we are dealing with a **glass**. A glass is fully out of equilibrium, its relaxation time is too large compared to our experimental time and **aging** (i.e. dynamics that depends on the initial time at which the glass was formed) is observed. Thus the cooling protocol to form a supercooled liquid depends on the trends of relaxation time τ_R and crystal nucleation time τ_N versus the temperature. $\tau_R(T)$ is monotonously increasing as T decreases while $\tau_N(T)$ diverges at T_m and $T = 0$ with a minimum in the middle (discussed more in depth in [4]). In experimental settings a **nonlinear cooling protocol** is adopted: at high temperatures cooling is faster, since τ_R is small (the liquid equilibrates quickly) and τ_N is big (there's no risk of forming a crystal since it would take long time), at low temperatures cooling is slower, since τ_R increases (need more time to equilibrate) and also τ_N does after having reached its minimum (this allows to slow down the cooling without formation of crystals).

Regardless of the cooling protocol, when the liquid is brought at a characteristic temperature, referred to as T_g, it cannot be equilibrated anymore since its relaxation time (viscosity) blows up: the **dynamic glass transition** takes place. By varying the temperature in a relative narrow range around T_g viscosity can increase by up to 14 decades. This dramatic growth is common to many liquids with very different microscopic structures, including polymers, as you can see in Fig. 1.1. The glass transition is one of the most interesting **open problems** in condensed matter physics and thus the definition of T_g is not sharp. One common way of addressing it is as the temperature where the relaxation time τ_R exceeds the experimental time t_{exp}:

$$\tau_R(T < T_g) > t_{exp} \tag{1.1}$$

Even though the definition of the transition is based on the experimental time, this is still a good definition because the increase in viscosity is so sharp that the dependence of T_g on t_{exp} is **really weak** (less than logarithmic). Still this definition lacks something, it's somewhat artificial, arbitrary. To tell if something interesting is going on we need to observe some **qualitative change** in the structure/dynamics of these materials.

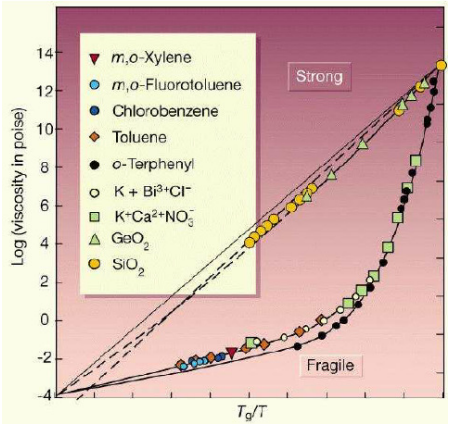

Figure 1.1: Angell's plot: Logarithm of the viscosity vs. rescaled inverse temperature for many substances. Taken from [5].

On the **static structure** side, there's no interesting change. Supercooled liquids around T_g, glasses under T_g and liquids well above T_g show the same qualitative structure of particles with only **minor changes while the viscosity varies of many decades**. This can be observed on the radial distribution function $g(r)$ which tells us what is the probability to find a particle at distance r from a certain focal particle:

$$g(r) = \frac{1}{N}\frac{1}{4\pi r^2 \rho}\left\langle \sum_i^N \sum_{j\neq i}^N \delta(r - r_{ij}) \right\rangle \tag{1.2}$$

where N is the total number of particles, ρ the density, r_{ij} the relative distance between particles i and j and $< . >$ represents the ensemble average over many realizations of the system. The radial distribution function is very good at distinguishing phases: a structured $g(r)$ with very sharp peaks that do not decay in space will point to an high degree of order in the phase, while weak, decaying peaks describe a disordered material. The $g(r)$ has no major changes at the glass transition and qualitatively looks like the one of a simple liquid. This is soon clear looking at Fig. 1.2, where the static structure factor $S(q)$ (a simple Fourier integral of the $g(r)$, more accessible experimentally) is plotted for the same glass-forming liquid at different temperatures:

The **dynamics** of particles, instead, shows a strong qualitative change. This is evident when looking at the incoherent intermediate scattering function:

$$F_s(\mathbf{q}, t) = \frac{1}{N}\sum_{k=1}^{N} \langle \delta\rho_k(\mathbf{q}, t)\delta\rho_k(\mathbf{q},0)\rangle \tag{1.3}$$

where $\delta\rho_k(\mathbf{q}, t) = \exp\left[-i\mathbf{q}\cdot\mathbf{r}_k(t)\right]$ is the Fourier transform of the density fluctuations of particle k at fixed "spatial frequency" $\mathbf{q}$. Being a correlation function,

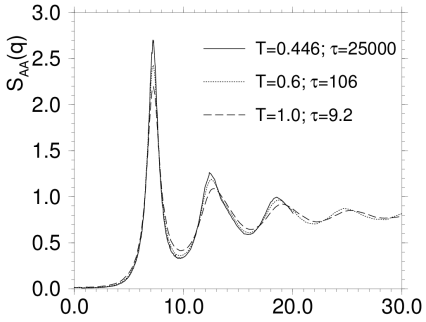

Figure 1.2: Static structure factor: $S(q)$ in a Lennard-Jones liquid at three temperatures with very different relaxation times τ. Taken from [4].

$F_s(\mathbf{q}, t)$ measures how correlation between density fluctuations of one particle decay in time. Looking at its evolution in temperature (Fig. 1.3) we can point out peculiarities of glassy relaxation dynamics. At high temperatures, after a short-time ballistic regime where particles move freely without interactions with the others, there's a dissipative regime where collisions between particles determine an exponentially decaying correlation. Approaching T_g, after the initial ballistic regime, there's a quick relaxation to a plateau (referred to as β relaxation) followed by a second, really slow, nonexpoential relaxation (α relaxation). This is what is called **two steps relaxation** and it's the footprint of glass transition: approaching it, the dynamics of particles changes **qualitatively** and cannot be described in terms of one time-scale anymore.

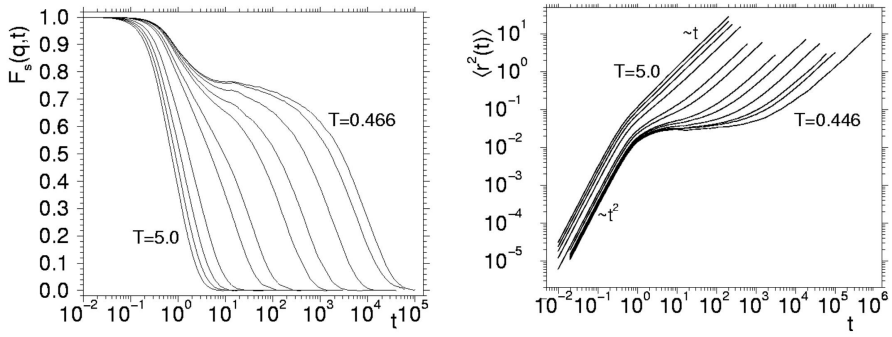

Figure 1.3: Two steps relaxation: *(left)* $F_s(\mathbf{q}, t)$ evaluated at the value of q where $S(q)$ has the main peak and at different temperatures. *(right)* MSD as a function of log-time. Computed on numerical simulations of a LJ glass-forming liquid. Taken from [4].

To shed more light on the dynamics in supercooled liquids we will look at the

mean square displacement of particles:

$$\left\langle r^2(t) \right\rangle = \frac{1}{N} \sum_{k=1}^{N} \left\langle ||\vec{x}_k(t) - \vec{x}_k(0)||^2 \right\rangle \tag{1.4}$$

As you can see in Fig. 1.3, at high temperatures, it shows the typical diffusion behavior with the initial ballistics regime in which $\langle r^2(t) \rangle \sim t^2$ followed by the diffusive regime $\langle r^2(t) \rangle \sim t$. Approaching T_g, the ballistic and the diffusive regime are separated by a plateau whose length increases decreasing the temperature that matches the one observed in the intermediate scattering function. The most spread interpretation of these curves is **the cage** picture: a tagged particle moves in a ballistic fashion for a very short time, then it gets trapped in the cage formed by its nearest neighbors and keeps vibrating inside it. After a long time, it can escape the cage and the diffusive motion is observed. The caging dynamics is a **collective behavior**: particles hops correspond to rearranging of different cages, indeed at random times, a number of cages is dismissed and new ones are formed. This is an high localized phenomenon in the sense that in a glassy liquid one can distinguish patches with high mobility (frequent rearranging) and patches with low mobility (purely vibrating particles). These patches are observed also in numerical simulations of glass-forming liquids (shown in Fig 1.4), they are self-correlated in time and studying their dynamic structure might be of fundamental importance for the understanding of the glass transition. This phenomenon is known as **dynamic heterogeneity**.

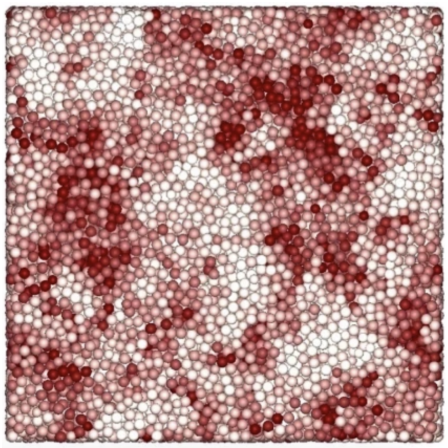

Figure 1.4: Dynamic heterogeneity: particles are colored according to the difference of their mobility from the mean. Red is above the mean (mobile particles), white below (vibrating particles). Take from [6].

To conclude this short review of super-cooled liquids phenomenology, one should note that the cage picture is just a **simplified description** of glassy dynamics. In

numerical simulations indeed, different behaviors are observed: particles can make jumps on different length-scales and also smooth transitions over long times are present. Nevertheless it is a useful picture because it relates the diverging viscosity of the fluid to the dynamics of its constituents.

1.2 Machine Learning for glassy liquids

The idea of using Machine Learning to attack the problem of glassy liquids is rather recent: the first work dates back to 2015 [2], and this line of research has continued [7][8][9] up to very recently [3]. These work aim mainly at leveraging the information contained in the static structure of the glassy systems to **predict the mobility of the particles**. They show that some weighted combinations of local expert features, like the $g(r)$ of each particle, are strongly correlated with the rearrangement dynamics of glassy systems. Thus they support the idea that a **structural order parameter** pointing at the glass transition should exist and should be a complex combination of these expert features. The major drawback of these works is that they rely on **simple ML technology** as SVMs [2] or trees (XGBoost) [10] and they perform a rather simplified task: binary classification of mobile and steady particles.

A parallel line of research [6] focuses on **unsupervised learning**, where instead of predicting mobilities, clustering methods are applied to distinguish different neighborhoods of particles. Mobility of different clusters are analyzed to confirm that the dynamics is strongly correlated with local static structure.

1.3 Graph Neural Networks

Most of the successes achieved by Machine Learning in the last decade rely on availability of large datasets and computational resources as well as on the ability of building architectures that are able to exploit intrinsic biases present in data. One of the most prominent examples is the success of Convolutional Neural Networks that, implementing locality and translational invariance, yielded impressive results on visual data. The biggest limitation of CNNs is their ability to handle only data lying on a regular grid, an issue which is solved by Graph Neural Networks.

GNNs are able to **reason on graph-based data** i.e. information is processed by these machines in the form of a graph were nodes represent objects and edges relation between them. Nodes and edges are endowed with features which can be scalar or multidimensional arrays and a forward pass of the GNN consists in updating recursively these features based on local rules (shown pictorially in Fig. 1.5) which are implemented thanks to **message passing routines** (explained in deep in Sec. 2) and that can adapt to different local geometries. This can be

seen simply as a **generalization to irregular grids of convolution kernels** used in CNNs. Thus, as in the case of CNNs, learning consists in modifying local update rules which usually are implemented through multi-layer perceptrons. So the learning process boils down to the update of MLPs weights. Thanks to this approach, GNNs learn how to update nodes and edges features based on their neighborhood independently of its specific structure. Thus they are able to **generalize to unseen graphs** with completely different geometries. In other words, GNNs are able to reason about a never-seen system because they're able to draw analogies by aligning relational structures as human mind does [11].

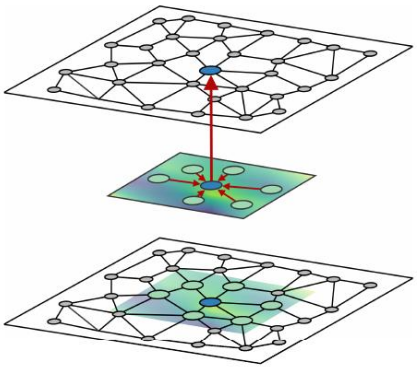

Figure 1.5: GCN: Generalization of continuous kernel to irregular geometries [12]. The central node feature is updated on the basis of a convolution of a continuous kernel with the node features of neighboring nodes.

The versatility of the graph structure has drawn a lot of attention to GNNs in the past 2-3 years. Applications range from **relational learning** [11], to **social networks analysis**, where one can classify parts of the graph [13] to **biochemistry**, for predicting proteins interactions [14], to **physics**, for quantum properties prediction [15][16] or for accelerating molecular dynamics simulations on meshes [17]. Some applications are present also in the field of **materials sciences** dealing mostly with crystalline materials. Our purpose is to apply this paradigm to amorphous materials where everything becomes harder because of the irregularity of the static structure.

1.4 State of the art

Recently, there has been a first attempt at using Graph Neural Networks on glasses [18], which **outperforms all previous methods** pertaining to the same task family.

The work focuses on the study of data coming from molecular dynamics simulations of a binary 80:20 Kob-Andersen-type Lennard-Jones mixture at different temperatures (relaxation times). The goal is the one of **predicting particles mobilities** on different timescales, starting **from the static structure** of the liquid, embedding its geometry in the graph network.

The training dataset is composed of 400 MD snaphsots of $N = 4096$ particles for each state point. From each snapshot a graph is built (Fig. 1.6a) taking particles as nodes and their type (0 for A and 1 for B) as **node feature**, two nodes are connected by an edge if their distance is less then a fixed threshold and the **feature of the directed edge** is the relative position of the nodes it is connecting. **Labels** are computed through the propensity approach: each particle is labeled with the average $\Delta r_i(t)$ computed on 30 trajectories ran with the same initial positions (the ones of the snapshot) and different initial velocities drawn from a Maxwell-Boltzmann distribution.

The task is the one of regressing the propensity of particles of type A. The GNN trained to do this has a rather simple **architecture** (Fig. 1.6c):

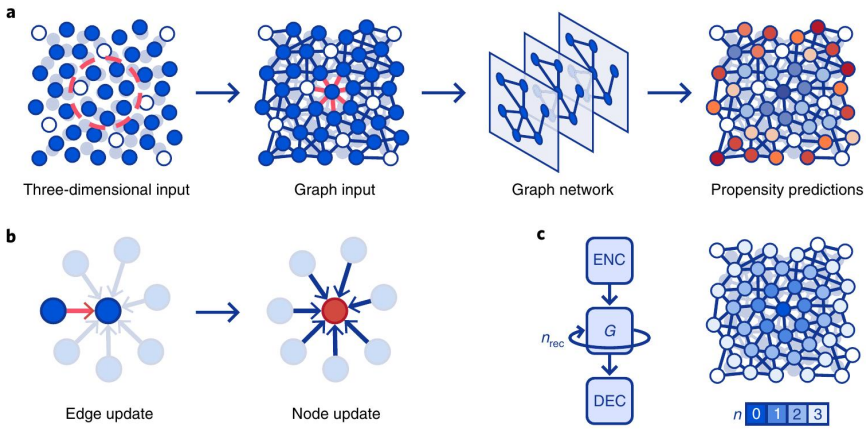

Figure 1.6: GNN for glasses: Essential steps involved in [18]. a) From the three-dimensional inputs, nodes at a distance of less than 2 LJ-units (defined in Sec. 2) are connected by two directed edges to form a graph. After processing, the network predicts propensities for each particle of type A b) Message passing routine: edge updates are based on the features of the given edge and of its associated nodes. Node updates are based on the features of the node and on the sum of its associated updated edge features. c) Structure of the GNN: encoder, n_{rec} applications of the message passing routine G and decoder. Each recurrence increases the shell of nodes contributing to the final value of a central node.

- **Encoder**: nodes and edges of the input graph are encoded independently with two two-layer multilayer perceptrons (MLP).

- **Recursive Message Passing**: nodes and edges are updated recursively thanks to a message passing routine (Fig. 1.6b): edge updates are based on the features of the given edge and of its associated nodes, concatenated and passed through an MLP. All edges are updated in parallel using the same MLP (edge MLP). Node updates are based on the features of the node and on the sum of its associated updated edge features, concatenated and passed through another MLP. Again, all nodes are updated in parallel using the same MLP (node MLP) which is different from the edge MLP. This routine allows to update all nodes and edges in the graph based on the same rule in the spirit of weight sharing of CNNs, where the same convolution kernel is used for the whole grid of data.

- **Decoder**: nodes and edges of the last recurrent layer are decoded independently with two two-layer MLP to regress towards propensity.

Also skip connections are used to stabilize learning and stochastic gradient descent is used in the learning process.

With this setting it's possible to outperform all previous methods, but this work represents more a **proof of concept** than a conclusive solution: they're still far away from reaching the upper bound for predictions correlation (see Suppl. Mat [18]) and there's a lot of room for improvement, both on the architecture which is rather simple and on the way the physical information is encoded in the ML task (labels and data features). In next sections we are going to characterize formally the architecture of the GNN and discuss in depth possible improvements.

1.5 Goals and perspectives

The goal of this work is primarily **understand and critically assess** the performance of state of the art GNN and ultimately pave the way for future **performances improvement** of the latter by exploiting information coming from expert features, designing new architectures or defining better measures of mobility to predict.

We explore mainly two directions:

- **Inclusion of expert features**: we introduce the $g_i(r) = \frac{1}{4\pi r^2} \sum_{j \neq i}^{N} \delta(r - r_{ij})$ or its dimensionally reduced version for each particle as node feature and consequently adapt the architecture to leverage the additional information provided to the GNN. This line aims at **bridging the gap** between old ML approaches in which **information-rich features** were combined with simple, less expressive models and new Deep-Learning approaches in which raw data is fed to **highly complex models**.

- **Definition of better measures of mobility**: in most of ML applications to glassy systems, measures of mobility used to compute the labels are really simple (e.g. propensity used in [18]). Here we study two different measures that are able to give more information on the trajectory itself and not only on the total displacement: p_{hop} introduced in [3] and *Wavelets transform* which is a new tool never used in this field.

We are not able to pair or beat the current state of the art, but our task is significantly harder, as explained in deep in next sections. We conclude that predicting single particle raw dynamics is an hard task for the GNN we have implemented and approaches like iso-configurational mean or spatial coarse-graining of the labels are needed to improve performances. Many advances could be done also on the **architecture** side, introducing **attention mechanism** or a new update routine, exploiting for example **generalized continuous convolutions** from [12]. Finally, once a very accurate model is built, due to high **interpretability** of GNNs, studying the learning dynamics and discerning the most relevant features for the model to learn will help to build **explainable** models and make advances in fundamental physics.

Chapter 2

Methods

2.1 Dataset: molecular dynamics simulations

The data we analyze in this work are obtained from molecular dynamics simulations. We simulate an **80:20 Kob-Andersen mixture** of $N = 64.000$ particles: 80% of them are the bigger A particles, 20% the smaller B ones. The interaction potential used for the simulations is the classic LJ potential, for two generic particles of types X and Y we have:

$$V_{XY}(r) = 4\varepsilon_{XY}\left(\left(\frac{\sigma_{XY}}{r}\right)^{12} - \left(\frac{\sigma_{XY}}{r}\right)^{6}\right) \tag{2.1}$$

with parameters

$$\varepsilon_{AA} = 1 \quad \varepsilon_{AB} = 1.5 \quad \varepsilon_{BB} = 0.5 \tag{2.2}$$

$$\sigma_{AA} = 1 \quad \sigma_{AB} = 0.8 \quad \sigma_{BB} = 0.88 \tag{2.3}$$

The potential is actually cut at $r = r_{cut}$ and put to 0, in the neighborhood of r_{cut} it's also smoothed in order to have vanishing first and second derivative in that point. The form of the smoothed LJ potential comes from the HOOMD library [19] used for the simulations. On the basis of the Lennard-Jones potential, we define dimensionless units and measure distances in units of σ_{AA}, time in units of $\tau = \sqrt{\frac{m\sigma_{AA}^2}{\varepsilon_{AA}}}$ and temperatures in units of $\frac{\varepsilon_{AA}}{k_B}$ where k_B is the Boltzmann constant. We use double precision, time steps of $dt = 0.0025$ LJ-time units and record frames once every 40 steps, corresponding to an inter-frame time step of 0.1 LJ-time units. The simulated system is composed of 64.000 particles in a simulation box of linear size $L = 37.64144$, thus the number density is $\rho = 1.2$ particles per unit volume.

We want to study the supercooled regime of the liquid, so we simulate the dynamics at one state point, $T = 0.43$ which is close to the mode coupling

temperature $T_{MCT} = 0.435$, the transition temperature predicted by Mode Coupling Theory [20] (not treated here since the purpose of the work is not theoretical). Indeed we observe the typical glassy behaviour in our simulations, see Fig. 2.1.

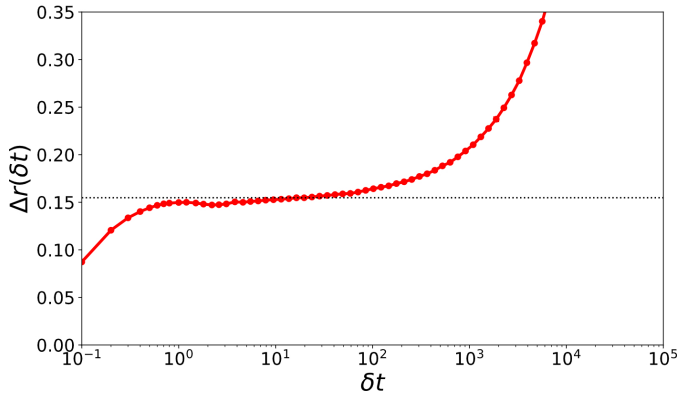

Figure 2.1: RMSD: Root mean square displacement of particles computed numerically on our simulations at temperature $T = 0.43$. It shows the typical glassy behavior (see Fig. 1.3). The black dashed line locates the plateau.

To prepare a sample at low temperature, a series of simulation steps are performed. At first, the system is equilibrated at high temperature $T \sim 1$: starting from random particles positions an NVT simulation with Nosé-Hoover thermostat is run. Then, to reach the $T_{target} = 0.43$, another NVT simulation with Maxwell-Boltzmann thermostat is run at $T \sim T_{target}$. The dynamics implemented in the M-B thermostat allows the system to thermalize quickly even though is less physical than N-H. So, after the thermalization, a final long NVT run with N-H thermostat is performed exactly at $T = T_{target}$. Once the system is equilibrated at T_{target} NVE simulations are run to record trajectories. NOTE: before starting NVE runs, we want the instantaneous energy, which in the NVT was fluctuating around the mean, to be exactly equal to the mean. To do so, particle velocities are rescaled by a constant quantity.

When simulating low temperature dynamics one has to check that the sample **has not crystallized**. One of the checks that have been done is looking at the time evolution of τ_α which is defined as the Δt for which the self intermediate scattering function is equal to $1/e$: $F_s(\mathbf{q}, \tau_\alpha) = e^{-1}$ evaluated at $\mathbf{q}$ s.t. $S(\mathbf{q})$ has its first maximum. This quantity describes the characteristic time of alpha relaxation and it depends on the level of super-cooling. It simply measures the time needed to have a fraction $1/e$ of particles that has not moved. τ_α is of order $6 * 10^3$ for particles A and $3 * 10^3$ for particles B for our system at $T = 0.43$. If parts of the system crystallized, it would increase by orders of magnitude so we check the time evolution of τ_α computed numerically on our simulated trajectories to make sure

that this kind of behavior is not observed. Fig. 2.2 shows a typical shape of τ_α vs time curve.

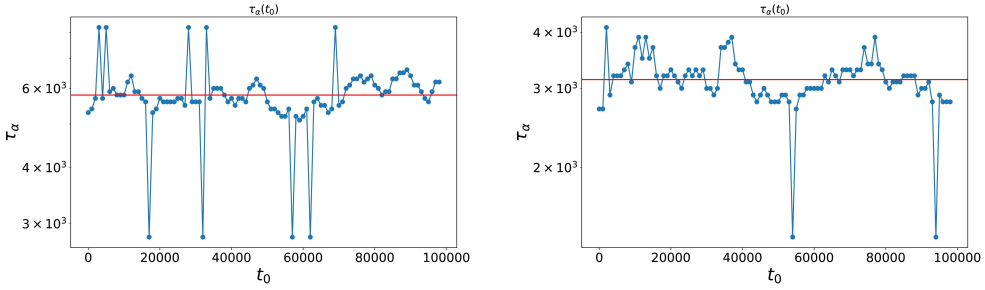

Figure 2.2: Tau alpha vs time: computed numerically on simulations at temperature $T = 0.43$, for particles A (left) and B(right)

To build the dataset, we simulate 8 separate copies of our system initialized differently in a random way. For each one the whole process described above is done independently, thus we get 8 instances of the same liquid thermalized at low temperature. Starting from these, we run NVE simulations to record trajectories. In order to **maximize the information contained in the dataset** and reduce the correlation between samples, for each system, we repeat multiple times the following procedure: we run an NVE simulation of duration $30\tau_\alpha$ to forget previous positions and record an NVE trajectory of duration $1\tau_\alpha$ to build features and labels. In this way we are sure that, between one sample and another one, most of the particles have moved as you can see in Fig. 2.3
Once we have a set of $1\tau_\alpha$-long independent trajectories we extract one sample from each of them as explained in the next sections.
In the end our **standard dataset** is made of 28 snapshots of $N = 64e3$ particles for **training** and 8 snapshots of the same size for **test**.

2.2 Graph: node and edge features

To feed the GNN we build one graph for each sample. Following the notation of [11], we define a graph as a 3-tuple $G = (\mathbf{u}, V, E)$ where $\mathbf{u}$ is a global attribute, $V = \{\mathbf{v}_i\}_{i=1,\dots,N^v}$ is the set of nodes with $\mathbf{v}_i$ as feature of node i and $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1,\dots,N^e}$ is the set of edges with $\mathbf{e}_k$ as feature of edge k, r_k receiver node and s_k sender. Building the graph means defining the set of nodes and directed edges, and attribute features to them.

Graph structure We take one snapshot from the middle of a $1\tau_\alpha$ trajectory and associate each particle to a node, two nodes are connected by a couple of directed

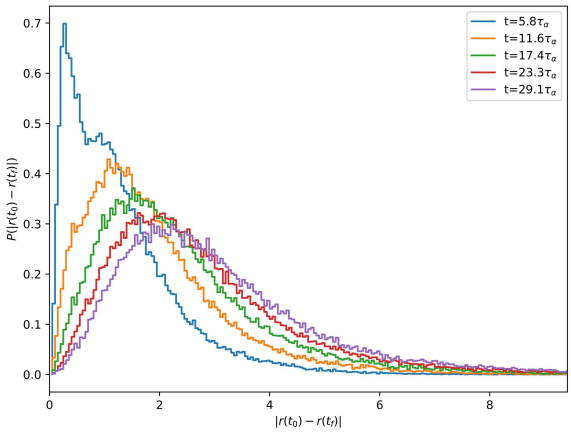

Figure 2.3: PDF of displacements: Probability density function (PDF) of displacements at different timescales at temperature $T = 0.43$. Computed numerically on a $30\tau_\alpha$ trajectory, building, for each timescale, an histogram from the displacements of all the particles in the system. For the curve related to $t = 5.8\tau_\alpha$ a bi-modal distribution is observed, this reflects the presence of two populations of particles, one which is caged and one which have hopped outside of the cage.

edges if the distance of the particle associated to them is less than a certain threshold:

$$(\mathbf{e}_k, r_k, s_k) \in E \iff |\vec{r}_{r_k} - \vec{r}_{s_k}| < r_{thr} \tag{2.4}$$

$r_{thr} = 1.5$ is chosen as the distance at which the $g(r)$ has the first minimum, i.e. particles are connected by edges if they're first neighbors.

Edge features Each edge is endowed with the distance between the particles it is connecting

$$e_k = |\vec{r}_{r_k} - \vec{r}_{s_k}| \tag{2.5}$$

Note: two nodes are connected by two directed edges with opposite direction and the same feature, it's a way of implementing un-directed edges.

Node features Various possibilities are explored in this work.

- **Particle types**: each node has a 1-dimensional feature T which is a binary variable: 0 if the particle is of type A and 1 if B.
- $\mathbf{g_i(r)}$: we feed the GNN with an expert feature which is known to strongly correlate with the dynamics of particles in supercooled regime [3]. Considering

a particle i we build a vector $\vec{G}(i)$ of 160 descriptors of the local structure around it as follows:

$$G_{Y,\alpha}(i) = \frac{1}{4\pi r_\alpha^2} \sum_{j\in\{Y\}} \mathbb{I}_{|\vec{r}_j - \vec{r}_i| \in [r_\alpha, r_{\alpha+1}]} \tag{2.6}$$

where Y denotes the type of particles (A,B) and $r_\alpha = \{0.70, 0.75, ..., 4.70\}$ is an 80-point binning of the distance (see Fig. 2.4). These descriptors are just a discretization of the $g_i(r)$, indeed $G_{Y,\alpha}(i)$ is equal to $g_{i,Y}(r_\alpha)$ up to a proportionality constant, where $g_{i,Y}(r) = \frac{1}{4\pi r^2} \sum_{j\in Y}^{N} \delta(r - r_{ij})$ is just the radial distribution function computed for particle i and types Y.

In the end each node is fed with 161 features:

$$\mathbf{v}_i = (T_i, \vec{G}_A(i), \vec{G}_B(i)) \tag{2.7}$$

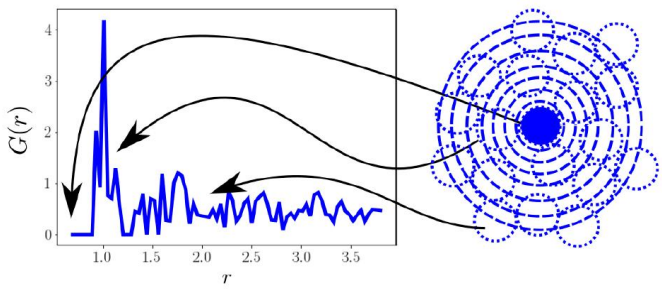

Figure 2.4: Descriptors: graphical representation of how descriptors are built. Numerically one has to count how many particles belongs to a shell at distance r_α from particle i and normalize this number by the surface of the shell.

- **Dimensionally reduced $\mathbf{g_i(r)}$**: we reduce the dimensionality of the descriptors in order to introduce less parameters in our model. Reducing the number of features present on the nodes means simplifying the encoder architecture. The main goal of this approach is reducing over-fitting. We perform a principal component analysis on a dataset $\mathcal{D} = \{(T_i^s, \vec{G}^s(i))\}$ where $s = 1, ...,28$ is the number of the train sample and $i = 0, ...,63999$ the particle number in each sample. So we fit the PCA model to the train dataset, in Fig. 2.5 you can see the cumulative explained variance as a function of the number principal components considered. Once the number of components is chosen (we try with $\{10,30,100\}$) all the $(T_i^s, \vec{G}^s(i))$ are projected on these components and their dimensionally reduced version is fed to the nodes rather than the complete one.

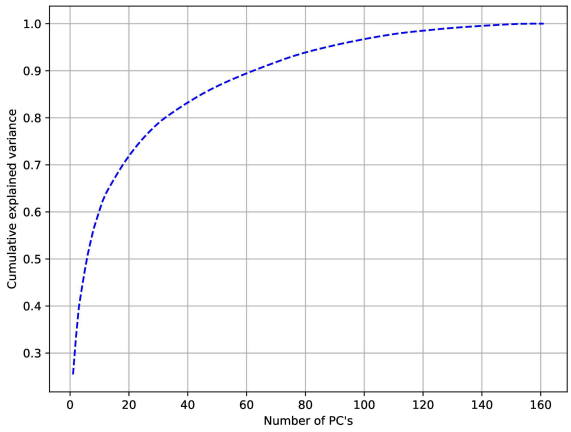

Figure 2.5: PCA of g(r): cumulative fraction of explained variance of the dataset as a function of the number of selected principal components. No knee is observed meaning that linear combinations of these features cannot explain the whole variance of data.

2.3 GNN architecture

The GNN architecture can be characterized formally following the notation of [11]. It's the same architecture as the one introduced in section 1.4 with a classic **encoder - recurrent update - decoder** structure.

The **encoder** encodes independently nodes and edges of the input graph with 2 two-layer MLPs with default size (64,64) and ReLu non-linearity: given that the initial graph is in the form $G^0 = (\mathbf{u}^0, V^0, E^0)$ the encoder will process its features as

$$\begin{aligned} \mathbf{u}^1 &= \mathbf{u}^0 \\ \mathbf{v}_i^1 &= \mathrm{Enc}^v(\mathbf{v}_i^0) = \mathrm{MLP}_{(64,64)}(\mathbf{v}_i^0) \\ \mathbf{e}_k^1 &= \mathrm{Enc}^e(\mathbf{e}_k^0) = \mathrm{MLP}_{(64,64)}(\mathbf{e}_k^0) \end{aligned} \tag{2.8}$$

Once the graph is encoded, its features are updated recursively based on what we will call **Graph Network (GN) block**, according to [11]. A GN block is made of update functions ϕ and aggregation functions ρ and it works in the following way:

1. Each edge feature is updated on the basis of its previous value, its associated nodes features and global features through an update function which is implemented as a two-layer MLP of default size (64,64) and ReLu non-linearity:

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) = \mathrm{MLP}_{(64,64)}(\mathbf{e}_k || \mathbf{v}_{r_k} || \mathbf{v}_{s_k} || \mathbf{u}) \tag{2.9}$$

where || represents concatenation.

2. Updated edges features are aggregated to update nodes. All the features of edges that are pointing at node i are aggregated into a single feature $\bar{\mathbf{e}}'_i$ that will be used for the update of node i feature:

$$\bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i) = \sum_{k:r_k=i} \mathbf{e}'_k \tag{2.10}$$

where $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i;k=1,..,N^e}$

3. Each node feature is updated on the basis of its previous value, the aggregated edges computed at the previous step and the global features. The update function is implemented as a two-layer MLP of default size (64,64) and ReLu non-linearity:

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) = \mathrm{MLP}_{(64,64)}(\bar{\mathbf{e}}'_i||\mathbf{v}_i||\mathbf{u}) \tag{2.11}$$

4. Updated edges and nodes features are aggregated separately to update the global features:

$$\begin{aligned} \bar{\mathbf{e}}' &= \rho^{e\to u}(E') = \sum_k \mathbf{e}'_k \\ \bar{\mathbf{v}}' &= \rho^{v\to u}(V') = \sum_i \mathbf{v}'_i \end{aligned} \tag{2.12}$$

where $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1,...,N^e}$ and $V' = \{\mathbf{v}'_i\}_{i=1,...,N^v}$.

5. Global features are updated on the basis of their previous value and the aggregated edges and nodes features. The update function is implemented as an identity function since in this work **global features are not used** (initially put to 0):

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) = \mathbb{I}(\mathbf{u}) \tag{2.13}$$

The GN block is applied **recursively** for a number of times $0 < n_{rec} < 7$. At each recurrence, node update functions are **shared** between nodes, same thing for edge update functions, in the spirit of weight sharing of CNNs filters. Thus at each recurrent step two MLPs are instantiated, one for node update and one for edge update and their weights are adjusted during the training procedure. Furthermore, at each recurrent step, **skip connections** are implemented by concatenating to current edge and node features the ones of the initially encoded graph.

At the end of the process, features are **decoded** in order to regress node features of particles A to their mobility. The decoder is implemented as a two hidden layers MLP with default size (64,64) and ReLu non-linearity plus one non-activated

single-layer perceptron for regression: given that the final graph is in the form $G^{n_{rec}} = (\mathbf{u}^{n_{rec}}, V^{n_{rec}}, E^{n_{rec}})$ the decoder will process its features as

$$\begin{aligned} \mathbf{u}^d &= \mathbf{u}^{n_{rec}} \\ \mathbf{v}_i^d &= \mathrm{Dec}^v(\mathbf{v}_i^{n_{rec}}) = \mathrm{MLP}_{(64,64,1)}(\mathbf{v}_i^{n_{rec}}) \\ \mathbf{e}_k^d &= \mathrm{Dec}^e(\mathbf{e}_k^{n_{rec}}) = \mathrm{MLP}_{(64,64,1)}(\mathbf{e}_k^{n_{rec}}) \end{aligned} \tag{2.14}$$

The GNN is implemented in **TensorFlow** following the work of [18].

2.4 Labels

Most of previous works on ML for glassy systems use very rough definitions of mobility to compute regression targets or classification labels. For example Bapts et al. [18] use the **propensity approach** introduced in section 1.4. This approach gives **little information about the true dynamics** of particles because it considers only the total displacement on a fixed timescale and averages it over the iso-configurational ensemble. Predicting the iso-configurational dynamics at a single particle scale is not equivalent to predicting the true dynamics, according to [1], while this might be true at larger scales. Indeed the iso-configurational mean averages out the initial velocities contribution to the dynamics and keeps just information coming from the static structure. Thus they perform a task which is rather simplified. We introduce labels that a) are able to **capture qualitative differences** in particle dynamics b) are not averaged over the iso-configurational ensemble so **describe the true dynamics**.

In the following we explore two measures of mobility to build the regression labels: p_{hop} and *Wavelets transform* of trajectories.

Phop As explained in section 1.1, the dynamics of particles in the supercooled regime is characterized by a long period of fluctuations around a given average position (the caging) and rare, sudden jumps that correspond to the escape from the cage. A useful measure, capable of detecting this kind of behaviour, is p_{hop}. Here we present the steps of computation of p_{hop} even though we do not go into much detail, since it was not the focus of the internship, but rather a tool. Following [3]:

$$p_{hop}(i,t) = \sqrt{\langle (\vec{r}_i - \langle \vec{r}_i \rangle_U)^2 \rangle_V^{1/2} \, \langle (\vec{r}_i - \langle \vec{r}_i \rangle_V)^2 \rangle_U^{1/2}} \tag{2.15}$$

for each particle i and all $t \in W$ where $W = [t_1, t_2]$ is the time window of the trajectory we are studying. Averages are performed over time intervals $U = [t_1, t]$ and $V = [t, t_2]$. One wants to separate the trajectory, distinguishing jumps from caged oscillations, so we compute $t^* = \mathrm{argmax}_{t \in W}(p_{hop}(i,t))$ which is the time

that best separates the trajectory $\vec{r}_i([t_1, t_2])$ into two sub-trajectories $\vec{r}_i([t_1, t^*])$ and $\vec{r}_i([t^*, t_2])$. We record $p^*_{hop}(i) = p_{hop}(i, t^*)$ and the process is iterated on each sub-trajectory. In this way, if the initial trajectory is made of oscillations inside the cage, the value of $p^*_{hop}(i)$ is small and corresponds to the cage's size, while if there's a jump, it corresponds to the amplitude of the jump. Iteration is interrupted when $p^*_{hop}(i) < p_c$ where p_c is taken as the value of the RMSD at the plateau (see Figure-2.1). In this way, only displacements larger than the RMSD plateau are considered jumps. Finally, to keep into account that jumps are not instantaneous, $p^*_{hop}(i)$ is assigned to all $p_{hop}(i, t)$ for $t \in [t^* - t_f, t^* + t_f]$ where $t_f = 5$ LJ time units.

For each $1\tau_\alpha$ simulation of the system, p_{hop} is computed for all the particles and one value is assigned at each frame (time instant) per particle. Then the central frame, the one from which the graph has been built, is taken and its p_{hop} values are the labels to regress to. We proceed in this way since we want to predict the mobility of the particles at the same time of the static structure observation.

In Fig. 2.6 the distribution (PDF) of values of p_{hop} computed on some samples of 64.000 particles is shown. It's clearly bi-modal around $p_c = 0.16$ since there are two populations of particles: steady and hopping ones. The most interesting events to predict are the largest jumps corresponding to the right-tail of the distribution.

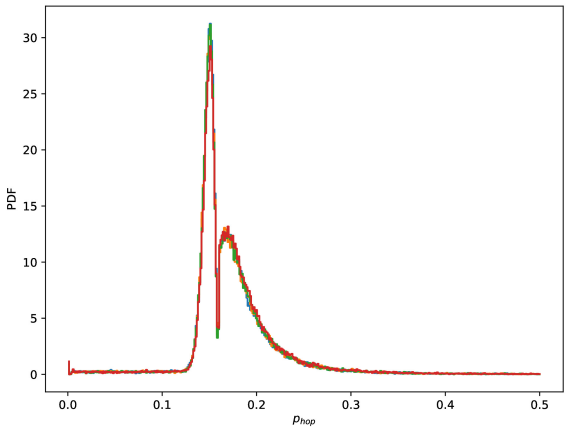

Figure 2.6: $\mathbf{p_{hop}}$ distribution: shown in different colors distributions computed on different samples

Wavelets transforms Wavelet transforms have been applied as a powerful tool in many fields, ranging from image and audio signal processing [21][22] to estimation of quantum properties of electronic systems [23]. They can be used to **detect variations (for example edges) in signals at different scales**. This is particularly useful for our task because not all the particles show a simple hopping/caging

behaviour, some of them move gradually on much longer timescales without having sudden jumps, as shown in Fig. 2.7. We perform a continuous wavelet transform

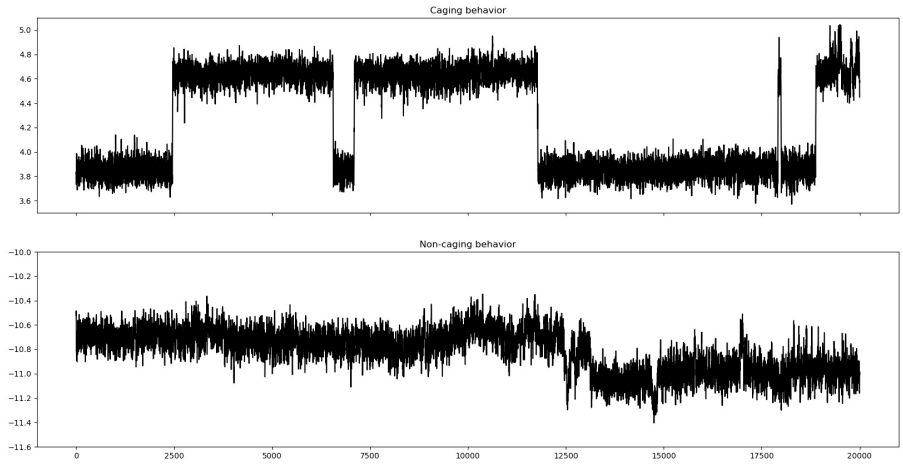

Figure 2.7: Non caging behavior: comparison between a trajectory that shows the typical caging behavior with oscillations in the cage and sudden jumps *(top)* and a trajectory with a slow transition *(bottom)*. Trajectories are obtained from MD simulations, plotting the position in the z direction of 2 different particles in the same liquid.

(CWT) of the trajectory of each particle i in each spatial direction d:

$$\mathbb{C}_i^d(a,t) = \int_{t_1}^{t_2} r_i^d(s) \frac{1}{\sqrt{a}} \overline{\psi\left(\frac{s-t}{a}\right)} \mathrm{ds} \tag{2.16}$$

where $[t_1, t_2]$ is the time window of the studied trajectory $\vec{r_i}(s) = (r_i^1(s), r_i^2(s), r_i^3(s))$. The CWT consist in convolving the time trajectory with a wavelet ψ stretched on various scales a and centered in all the time instants of the trajectory t (in this way each frame has a label). Here we chose as form of the wavelet $\psi(t) = -Cte^{-t^2}$ where C is a positive normalization constant and scales: $\vec{a}_{frames} =$ [50,100,200,400,800,1600,3200] frames which, expressed in LJ-time units are $\vec{a}_{LJ} =$ [5,10,20,40,80,160,320] since each simulation frame is printed every 0.1 LJ-time unit. Once the transform in each direction is computed, the label of particle i at frame t is computed as:

$$\vec{m}_i(t) = \begin{bmatrix} \sqrt{\sum_{d=1}^3 \mathbb{C}_i^d(a_0,t)^2} \\ \vdots \\ \sqrt{\sum_{d=1}^3 \mathbb{C}_i^d(a_{n_s},t)^2} \end{bmatrix} \tag{2.17}$$

where n_s is the number of scales used for the transform. The choice of the scales is done empirically in order to detect hops on different timescales and avoid boundary effects on $1\tau_\alpha$ trajectories. In Fig. 2.8 a typical trajectory with CWT coefficients is shown.

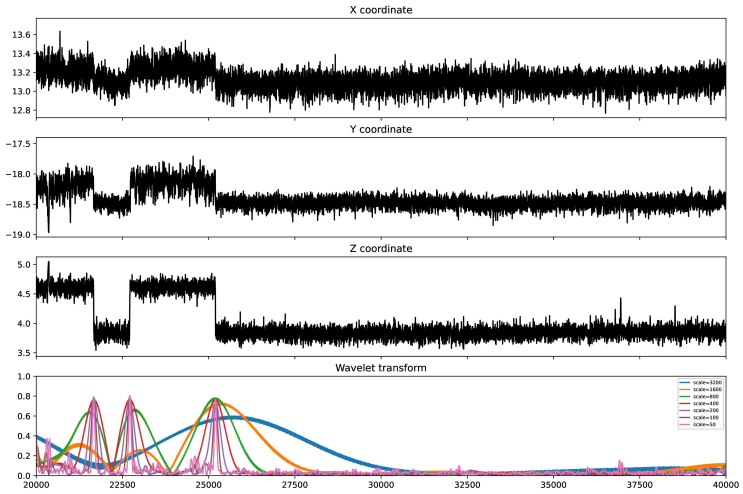

Figure 2.8: CWT: particle position is in LJ-units, time in frame number ($10 *$ LJ-time unit)

In the actual implementation, the convolution is computed for every particle just in the central frame of the $1\tau_\alpha$ trajectory and a multivariate label is assigned to each particle. This approach allows to **study the mobility of the particle at the very time of the static structure observation and at previous and later times** since the convolution has 'memory' of the rest of the trajectory. Finally this measure of mobility respects the **time reversal symmetry** of the equilibrium dynamics of supercooled liquids by construction, this is clear also qualitatively on similar jumps with opposite direction.

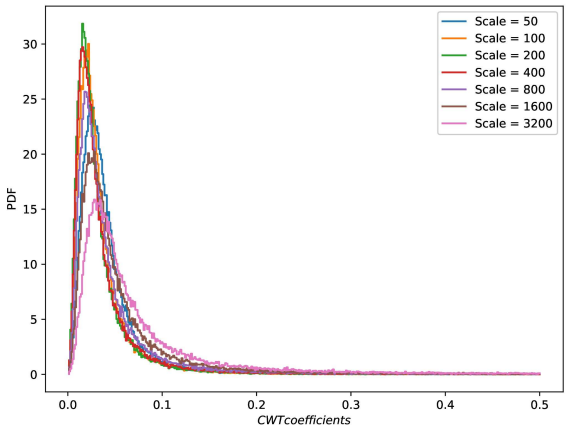

Figure 2.9: CWT distribution: PDF of wavelets coefficients computed on one sample with 64.000 values. Scales are expressed in number of frames.

Fig. 2.9 shows the typical PDF of CWT coefficients for different scales. Most of the information is in the tail of the distributions since large values are recorded in correspondence of significant displacements. This makes the task of predicting these values hard since we are dealing with rare events.

2.5 Training procedure

Simple regression This is the case of mono dimensional labels built from p_{hop} or single scale of CWT. The training procedure is performed through the **minimization of the $\mathcal{L}_2$-norm** of the error between predicted and true mobility of particles A:

$$L_{\mathcal{L}_2} = \sum_{i \in A} (t_i - m_i)^2 \tag{2.18}$$

Multivariate regression This is the case of multidimensional labels built from multiple scales of CWT. The training procedure consists in minimizing the global $\mathcal{L}_2$-loss which is obtained as the average of the loss in each dimension:

$$L_{\mathcal{L}_2}^{glob} = \frac{1}{n_s} \sum_{k=1}^{n_s} \sum_{i \in A} (t_{i,k} - m_{i.k})^2 \tag{2.19}$$

where the first sum is over the CWT scales and the second one over nodes (particles). $t_{i,k}$ is the true CWT at the k-th scale for particle i, $m_{i,k}$ is the corresponding predicted one.

Re-weighting the $\mathcal{L}_2$-loss Results obtained in the case of simple and multivariate $\mathcal{L}_2$ regression show that the GNN is not able to predict rare events, as pointed out in sections 3.1-3.2. Hence the need for the introduction of a re-weighting of the loss in order to take into account the real distribution of the labels. **Importance weighting** is performed over nodes: node i is associated with a weight w_i determined by the probability of occurrence of its target mobility in the training set ($p(t_i)$). This quantity is computed empirically as an histogram, through binning of the possible values. See for example Fig. 2.6. Weights are defined as:

$$w_i = \frac{1}{p(t_i)^\alpha} \tag{2.20}$$

and depend on the parameter $\alpha \in [0,1]$ which determines the relevance of the re-weighting and is fixed empirically. The training procedure consists in minimizing the loss:

$$L_w = \sum_{i \in A} w_i (t_i - m_i)^2 \tag{2.21}$$

In this way the largest values of mobility, which are the less frequent, are given a larger importance in the regression procedure.

In all the cases regression is performed on **single particle quantities**, no coarsening is applied to the last layer of the GNN and each particle is associated to a predicted mobility. **Early stopping** is implemented and the model with the highest accuracy on the test dataset is selected. The minimization is done with initial learning rate of 10^{-4}, gradient-clipping and **Adam optimizer**. Training was run on GPGPUs of lab-IA cluster at LISN of Paris-Saclay. For simple regression the **evaluation metric** used is the Pearson correlation coefficient ρ between predicted mobility m_i and true mobility t_i of type A particles:

$$\rho = \frac{\sum_{i=0}^{N_A}(m_i - \bar{m})(t_i - \bar{t})}{\sqrt{\sum_{i=0}^{N_A}(m_i - \bar{m})^2}\sqrt{\sum_{i=0}^{N_A}(t_i - \bar{t})^2}} \tag{2.22}$$

For multivariate regression, a different correlation coefficient is computed for each component of the output (CWT scale):

$$\rho_k = \frac{\sum_{i=0}^{N_A}(m_{i,k} - \bar{m}_k)(t_{i,k} - \bar{t}_k)}{\sqrt{\sum_{i=0}^{N_A}(m_{i,k} - \bar{m}_k)^2}\sqrt{\sum_{i=0}^{N_A}(t_{i,k} - \bar{t}_k)^2}} \tag{2.23}$$

with $k = 1, ..., n_s$.

Chapter 3

Results

The training of the GNN is performed with ground-truth built from the two measures of mobility discussed above: p_{hop} and CWT. For the first one, a full architecture search is performed, together with the exploration of different node features and dimensions of the dataset. For the second one, the work is still preliminary, but, keeping the architecture and the input features fixed, both simple regression and multivariate regression is performed, yielding same performances. In all the cases accuracy is rather low and GNN is able to predict only the most frequent value of the target. For this reason re-weighting of the loss is introduced, but no significant improvement is observed. We conclude that the level of randomness in our labels, due to initial conditions, is too relevant and iso-configurational mean / coarse-graining of degrees of freedom should be introduced.

3.1 Phop

3.1.1 Full g(r)

Architecture search We explore different sizes of the MLPs = {(8,8), (16,16), (32,32), (64,64)} and different numbers of recurrences $n_{rec} = \{0,1,2,3,4,5\}$. Figure 3.1 shows the results. Error-bars are obtained by testing the trained model on 4 subsets of the test-dataset and computing standard deviation of the test correlations. The case $n_{rec} = 0$ corresponds to a model in which only encoder and decoder are present, so no information about the geometry is present. Increasing the number of recurrences means increasing the maximum distance at which one node "feels" the others.

Training dynamics Training is performed as described in section 2.3, results are recorded every 100 training steps which correspond to ~ 3.6 epochs on the dataset. Early stopping is implemented checking at each recording step that the $\mathcal{L}_2$-loss on

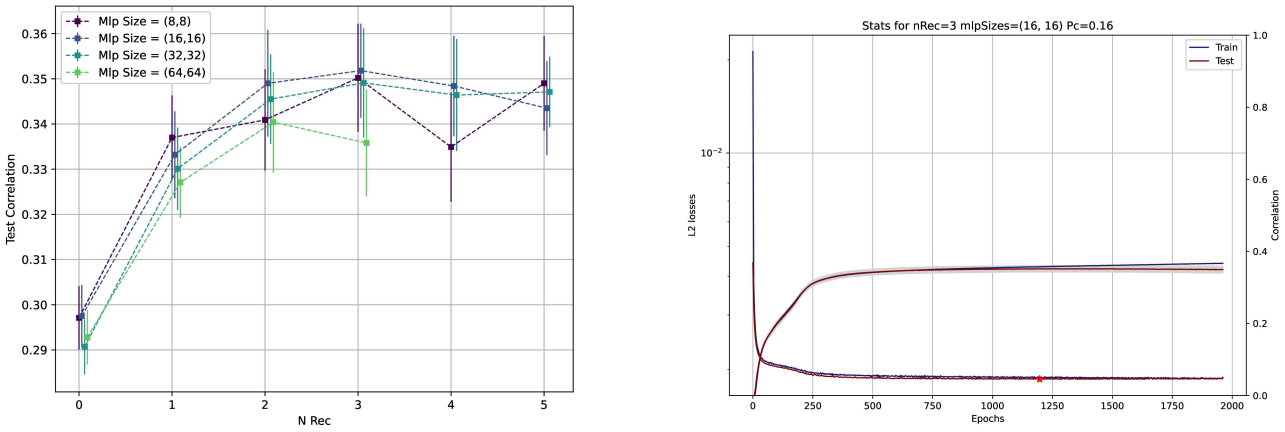

Figure 3.1: (left)**Architecture search**: The best correlation is reached for $n_{rec} = 3$ and MLP size (16,16) and is $\rho = 0.35 \pm 0.01$. For MLP size (64,64) it wasn't possible to train more than 3 recurrences due to hardware limitations (training was performed on NVIDIA Tesla V100 GPUs with 32 GiB of RAM). (right) **Training curve**: Training of the GNN with $n_{rec} = 3$ and MLP size (16,16). In the plot you have L2 losses in log-scale (monotonously decreasing curves), and Person correlation coefficients (monotonously increasing curves) versus training epochs. In grey, the standard deviation of the test correlation, the red star signals the minimum of the test $\mathcal{L}_2$-loss. The split between train and test correlation signals the kick in of over-fitting, more complex models start over-fitting the dataset sooner.

the test set is not higher than the previous n_{tol} ones. The best tolerance interval is found empirically for this task and is $n_{tol} = 100$ recording steps. Figure 3.1 shows a typical training curve.

Learning curve The best performing models ($n_{rec} = 3$) are chosen to explore the dataset dimension. The learning curve (Fig. 3.2) shows that **increasing the dataset size doesn't lead to major increase in accuracy of predictions**.

Predictions Here we present the predictions of the GNN with $n_{rec} = 3$ and MLP size (16,16). Both the scatter-plot of predicted labels vs. ground-truth and the PDF of the predicted labels suggest that re-weighting of the loss function is needed. The **GNN predicts only values around the mean of the ground-truth**, this is clear by looking at Fig. 3.3 where PDF of predicted and true labels are compared, plotted on a log scale.

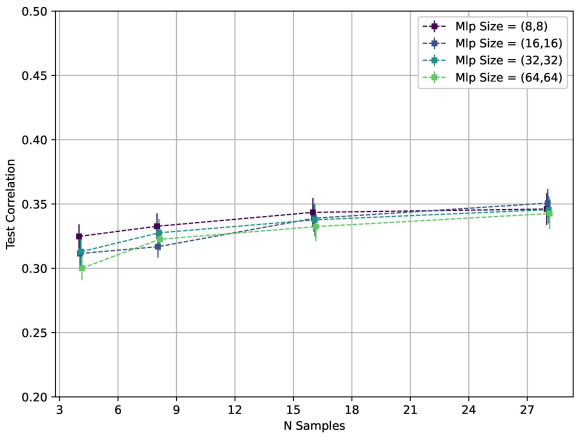

Figure 3.2: Learning curve: the training dataset is down-sampled at {4,8,16,28} samples and training performed as described before.

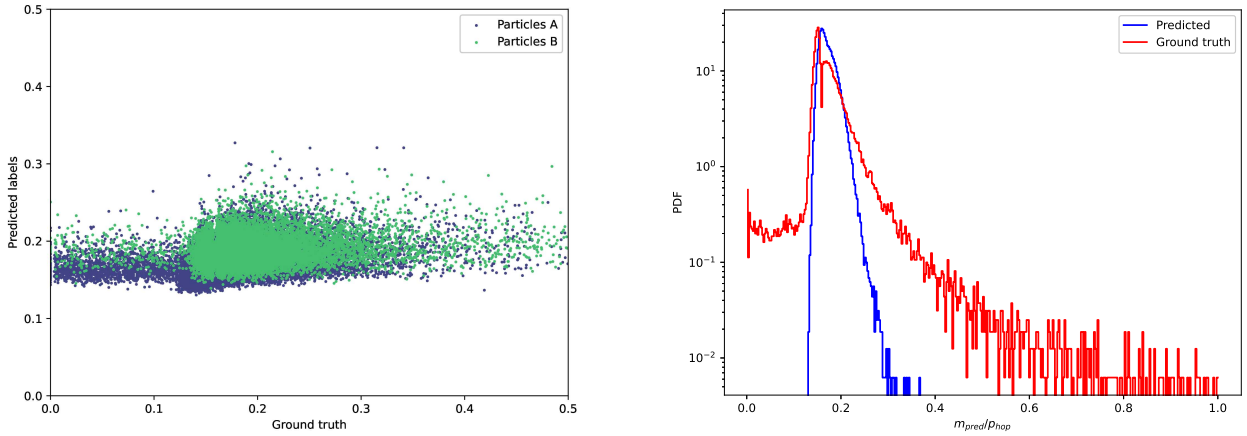

Figure 3.3: (left)**Scatter plot**: predictions vs. ground-truth for one sample of the test-set. (right) **PDF of predicted/true labels**: the GNN is not able to predict the tails.

3.1.2 PCA and particle types

Dimensionally reduced g(r) We feed the GNN with graphs whose node features have dimension 10, 30, 100. We try to keep as much information as possible still reducing the complexity of the model. Less node features means less weights to learn in the first layer of the Encoder. Indeed simpler models over-fit the dataset

slower and **between 30 and 161 features the performance seems to keep almost constant** (see Fig. 3.4).
Particle types The simplest case is the one with just one feature on each node, following the work of [18]. In the observed range of recurrences, **adding features always increases correlation**.

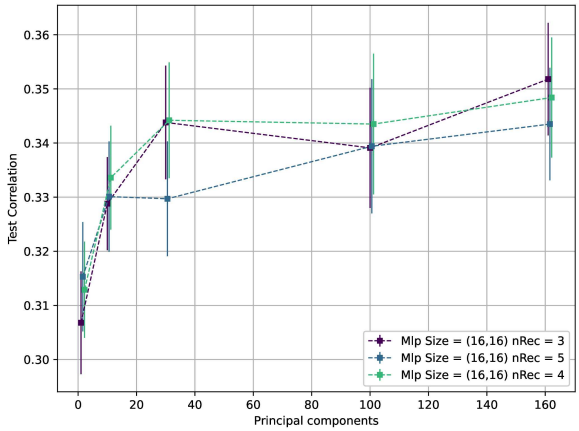

Figure 3.4: Node features: 1 component means only particle type, 161 full g(r) + particle type. Between 30 components ($> 70\%$ of explained variance), 100 components ($> 90\%$ of explained variance) and full g(r) there's no significant increase in performance.

3.2 Wavelets

We perform **simple regression** of node features considering only one scale per training and also **multivariate regression** i.e. we regress node features to labels that have dimension $d = n_s$. In the second case the architecture is modified since the last layer of the decoder is composed of d neurons.

Scales	Simple Reg	Multi Reg
80	$\rho = 0.12 \pm 0.06$	$\rho = 0.12 \pm 0.06$
160	$\rho = 0.13 \pm 0.05$	$\rho = 0.13 \pm 0.06$
320	$\rho = 0.15 \pm 0.07$	$\rho = 0.14 \pm 0.06$

Table 3.1: some training results for $n_{rec} = 3$ and MLP size (16,16). Scales are expressed in LJ-time units.

This is still preliminary work, but prediction analysis (Fig 3.5) shows that the difficulty of the task might come, as in the case of p_{hop}, from the difficulty of predicting rare events.

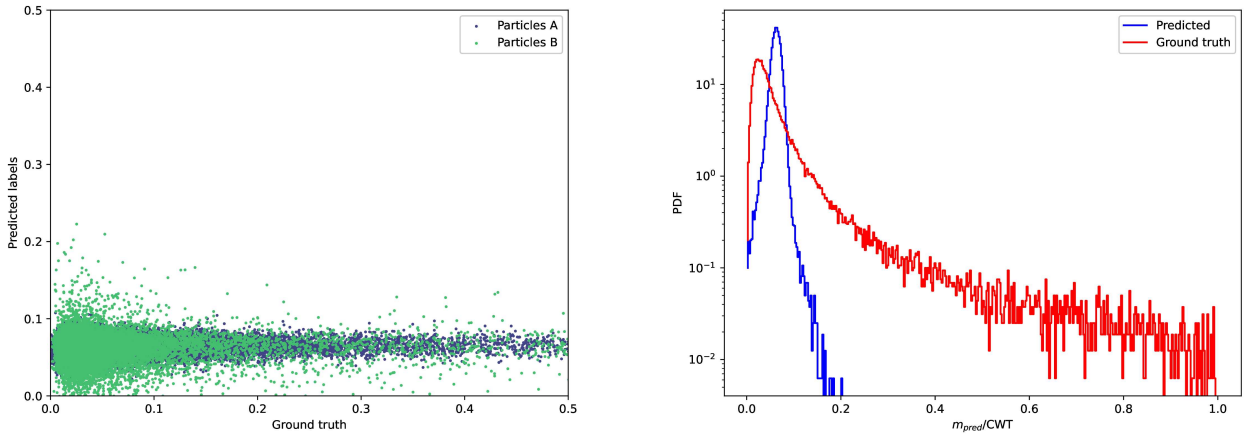

Figure 3.5: (left)**Scatter plot**: predictions vs. ground-truth for one sample of the test-set. (right) **PDF of predicted/true labels**: the GNN is able to predict only the average of the ground truth (with minor corrections) as in the case of p_{hop}.

3.3 Re-weighted loss

Importance weighting is performed on simple regression tasks i.e. predicting p_{hop} or single scale wavelet transform. Here we present results obtained with p_{hop}.

Starting form the configuration which yields the best results in the simple $\mathcal{L}_2$-loss minimization, weights are introduced according to section 2.5 and the parameter α is tuned empirically to match the ground truth distribution. Figure 3.6 shows results obtained for $\alpha = 0.5$: accuracy is still low and, even though the ground truth distribution is matched, the scatter plot of prediction vs. ground-truth makes evident that the GNN is not learning. Similar behavior is observed when simple regression is performed with importance weighting on single scale CWT.

We conclude that re-weighting alone is not sufficient to improve the learning because, regardless of their distribution, labels have an intrinsic random component which is relevant. This makes their prediction a difficult task.

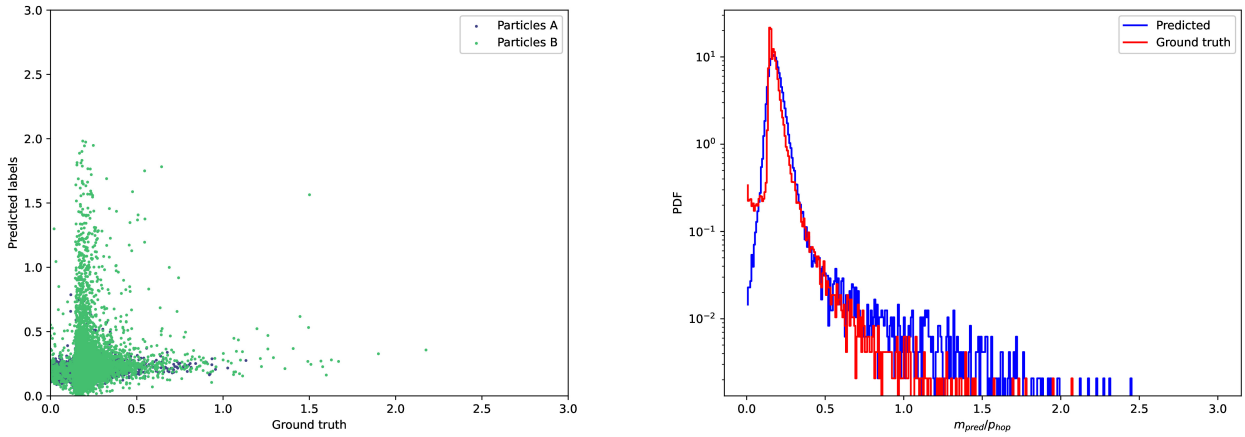

Figure 3.6: (left)**Scatter plot**: predictions vs. ground-truth for one sample of the test-set. (right) **PDF of predicted/true labels**: the GNN is able to predict also larger values of mobility but there's no correlation with ground truth so it is not learning.

Chapter 4

Discussion

Our results are partial, we're far from pairing or beating the current state of the art and there are still a lot of improvements to be done, but, as suggested by Battaglia et al. in [11], we believe that the path of including prior knowledge into complex and expressive architectures without disregarding expert features is the way to reach big improvements in ML in general. Still it is clear that **deeper re-thinking of the GNN architecture** is needed to leverage the additional information introduced by these features.

The analysis we have performed showed us that our labels are hard to predict because of the high level of randomness which is intrinsic in the glassy dynamics. Most of the previous works belonging to this line of research, included the one from Bapts et al.[18], focus on the prediction of the propensities which are averaged quantities that keep into account only the structural contribution to particles dynamics. They don't investigate the connection between propensities and true dynamics. We are trying to predict directly the true dynamics at single particle scale starting from the static structure and without passing through the iso-configurational averaged quantities. This is an hard task in fact, according to Berthier et al. in [1], the **dominant single-particle dynamical fluctuations are intrinsically dynamical**, and not linked with liquid structure. We observe this also in our system following their approach: we compare raw labels (for example CWT at a fixed scale) with the iso-configurational averaged ones. The result is the same (Fig 4.1): raw labels have a low correlation with the averaged ones meaning that studying the iso-configurational dynamics doesn't give us enough information on the true dynamics. Conversely, **predicting the true dynamics starting only from structural information is not feasible at a single particle level**.

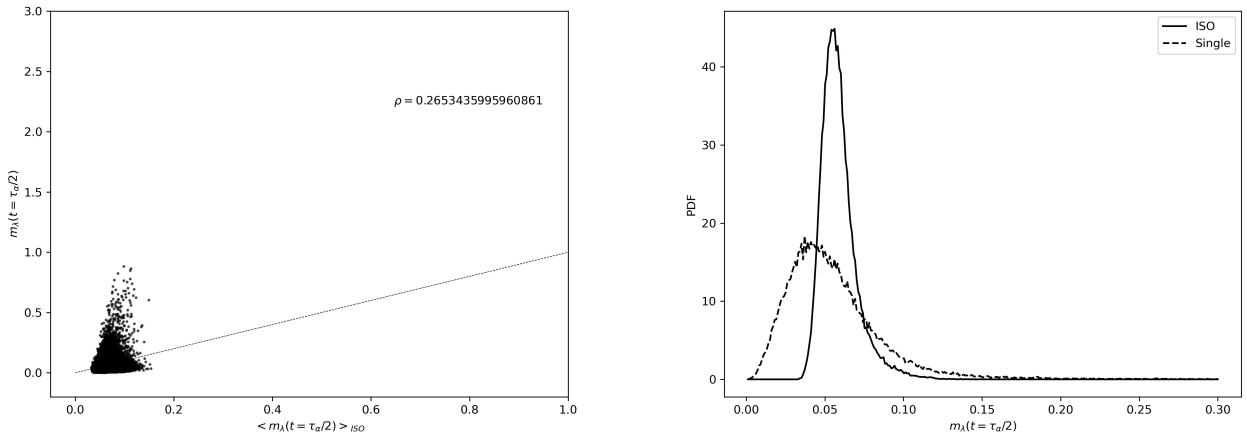

Figure 4.1: Comparison between raw labels and averaged ones. The latter are obtained as an average over 30 MD simulations starting from the same initial spatial structure and randomized velocities. Labels chosen here are CWT on one scale. (left)**Scatter plot**: raw labels versus iso-configurational ones evaluated at $t = \tau_\alpha/2$ where $t = 0$ is the initial configuration (right) **PDF of raw/iso labels**: raw labels distribution is more spread as expected.

Starting from the observations of [1] we conclude that we should **look at the dynamics on a coarser scale** if we want to be able to improve our predictions and clarify the link between structure and glassy dynamics. This could be done through a pooling procedure applied on the last layer of the GNN.

Once good results are achieved with the current architecture, further improvements would consist in adding **attention mechanism** [14] on top of it or **rethinking the architecture** using completely different routines to update nodes and edges features. Self-attention mechanisms would allow to perform updates of nodes features based on weighted averages of the neighborhood, thus revealing which neighbors are fundamental for the update of one node. Different update routines, like the one based on **generalized continuous convolutions** [12], would exploit more efficiently the spatial information fed to the GNN and it would learn some continuous convolution kernel that can be analyzed to extract physical information from it.

Finally, an interesting direction would be the one of building **explainable** models. The machine-learned output of the (successful) GNN, by definition, depends solely on the atomic structure of the sample. It is thus a machine-learned **structural order parameter**. However, it is hard to interpret and does not constitute, by itself, a progress in basic science. What would constitute such a progress would be an explainable order parameter. Many possibilities open up in

this sense: **knowledge distillation** i.e. analyze the trained neural network layers to check their correlation with a predefined list of hand-crafted physical quantities, building decision trees or symbolic meta-models which are explainable by design and so on. In general the issue of explainability is of central interest nowadays in machine learning and working on it would bring significant contribution for all the community.

Bibliography

[1] Ludovic Berthier and Robert L. Jack. «Structure and dynamics in glass-formers: predictability at large length scales». en. In: *Physical Review E* 76.4 (Oct. 2007). arXiv: 0706.1044, p. 041509. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.76.041509. URL: http://arxiv.org/abs/0706.1044 (visited on 02/17/2021) (cit. on pp. 1, 18, 30, 31).

[2] Samuel S. Schoenholz, Ekin D. Cubuk, Daniel M. Sussman, Efthimios Kaxiras, and Andrea J. Liu. «A structural approach to relaxation in glassy liquids». en. In: *arXiv:1506.07772 [cond-mat]* (Nov. 2015). arXiv: 1506.07772. URL: http://arxiv.org/abs/1506.07772 (visited on 02/17/2021) (cit. on pp. 1, 6).

[3] François P. Landes, Giulio Biroli, Olivier Dauchot, Andrea J. Liu, and David R. Reichman. «Attractive versus truncated repulsive supercooled liquids: The dynamics is encoded in the pair correlation function». en. In: *Physical Review E* 101.1 (Jan. 2020). arXiv: 1906.01103, p. 010602. ISSN: 2470-0045, 2470-0053. DOI: 10.1103/PhysRevE.101.010602. URL: http://arxiv.org/abs/1906.01103 (visited on 02/17/2021) (cit. on pp. 1, 6, 10, 14, 18).

[4] Andrea Cavagna. «Supercooled Liquids for Pedestrians». en. In: *Physics Reports* 476.4-6 (June 2009). arXiv: 0903.4264, pp. 51–124. ISSN: 03701573. DOI: 10.1016/j.physrep.2009.03.003. URL: http://arxiv.org/abs/0903.4264 (visited on 02/17/2021) (cit. on pp. 1, 2, 4).

[5] Francesco Arceri, François P. Landes, Ludovic Berthier, and Giulio Biroli. «Glasses and aging: A Statistical Mechanics Perspective». en. In: *arXiv:2006.09725 [cond-mat]* (Oct. 2020). arXiv: 2006.09725. URL: http://arxiv.org/abs/2006.09725 (visited on 02/17/2021) (cit. on p. 3).

[6] Emanuele Boattini, Susana Marín-Aguilar, Saheli Mitra, Giuseppe Foffi, Frank Smallenburg, and Laura Filion. «Autonomously revealing hidden local structures in supercooled liquids». en. In: *Nature Communications* 11.1 (Dec. 2020), p. 5479. ISSN: 2041-1723. DOI: 10.1038/s41467-020-19286-8. URL: http://www.nature.com/articles/s41467-020-19286-8 (visited on 05/19/2021) (cit. on pp. 5, 6).

[7] E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu. «Identifying Structural Flow Defects in Disordered Solids Using Machine-Learning Methods». In: *Physical Review Letters* 114.10 (2015). ISSN: 1079-7114. DOI: 10.1103/physrevlett.114.108001. URL: http://dx.doi.org/10.1103/PhysRevLett.114.108001 (cit. on p. 6).

[8] Ekin D. Cubuk, Samuel S. Schoenholz, Efthimios Kaxiras, and Andrea J. Liu. «Structural Properties of Defects in Glassy Liquids». In: *The Journal of Physical Chemistry B* 120.26 (2016). PMID: 27092716, pp. 6139–6146. DOI: 10.1021/acs.jpcb.6b02144. eprint: https://doi.org/10.1021/acs.jpcb.6b02144. URL: https://doi.org/10.1021/acs.jpcb.6b02144 (cit. on p. 6).

[9] Daniel M. Sussman, Samuel S. Schoenholz, Ekin D. Cubuk, and Andrea J. Liu. «Disconnecting structure and dynamics in glassy thin films». In: *Proceedings of the National Academy of Sciences* 114.40 (2017), pp. 10601–10605. ISSN: 1091-6490. DOI: 10.1073/pnas.1703927114. URL: http://dx.doi.org/10.1073/pnas.1703927114 (cit. on p. 6).

[10] Qi Wang, Jun Ding, Longfei Zhang, Evgeny Podryabinkin, Alexander Shapeev, and Evan Ma. «Predicting the propensity for thermally activated β events in metallic glasses via interpretable machine learning». en. In: *npj Computational Materials* 6.1 (Dec. 2020), p. 194. ISSN: 2057-3960. DOI: 10.1038/s41524-020-00467-4. URL: http://www.nature.com/articles/s41524-020-00467-4 (visited on 06/18/2021) (cit. on p. 6).

[11] Peter W. Battaglia et al. «Relational inductive biases, deep learning, and graph networks». en. In: *arXiv:1806.01261 [cs, stat]* (Oct. 2018). arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261 (visited on 03/26/2021) (cit. on pp. 7, 13, 16, 30).

[12] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Muller. «SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels». en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 869–877. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00097. URL: https://ieeexplore.ieee.org/document/8578195/ (visited on 05/17/2021) (cit. on pp. 7, 10, 31).

[13] Hongyang Gao and Shuiwang Ji. «Graph U-Nets». en. In: *arXiv:1905.05178 [cs, stat]* (May 2019). arXiv: 1905.05178. URL: http://arxiv.org/abs/1905.05178 (visited on 05/03/2021) (cit. on p. 7).

[14] Petar Veličkovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lió, and Yoshua Bengio. «Graph Attention Networks». en. In: *arXiv:1710.10903 [cs, stat]* (Feb. 2018). arXiv: 1710.10903. URL: http://arxiv.org/abs/1710.10903 (visited on 02/17/2021) (cit. on pp. 7, 31).

[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. «Neural Message Passing for Quantum Chemistry». en. In: *arXiv:1704.01212 [cs]* (June 2017). arXiv: 1704.01212. URL: http://arxiv.org/abs/1704.01212 (visited on 03/26/2021) (cit. on p. 7).

[16] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. «Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals». en. In: *Chemistry of Materials* 31.9 (May 2019), pp. 3564–3572. ISSN: 0897-4756, 1520-5002. DOI: 10.1021/acs.chemmater.9b01294. URL: https://pubs.acs.org/doi/10.1021/acs.chemmater.9b01294 (visited on 02/17/2021) (cit. on p. 7).

[17] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. «Learning Mesh-Based Simulation with Graph Networks». en. In: *arXiv:2010.03409 [cs]* (Jan. 2021). arXiv: 2010.03409. URL: http://arxiv.org/abs/2010.03409 (visited on 05/03/2021) (cit. on p. 7).

[18] V. Bapst et al. «Unveiling the predictive power of static structure in glassy systems». en. In: *Nature Physics* 16.4 (Apr. 2020), pp. 448–454. ISSN: 1745-2473, 1745-2481. DOI: 10.1038/s41567-020-0842-8. URL: http://www.nature.com/articles/s41567-020-0842-8 (visited on 02/17/2021) (cit. on pp. 7–10, 18, 27, 30).

[19] Joshua A. Anderson, Jens Glaser, and Sharon C. Glotzer. «HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations». In: *Computational Materials Science* 173 (2020), p. 109363. ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2019.109363. URL: https://www.sciencedirect.com/science/article/pii/S0927025619306627 (cit. on p. 11).

[20] Shankar P. Das. «Mode-coupling theory and the glass transition in supercooled liquids». In: *Rev. Mod. Phys.* 76 (3 Oct. 2004), pp. 785–851. DOI: 10.1103/RevModPhys.76.785. URL: https://link.aps.org/doi/10.1103/RevModPhys.76.785 (cit. on p. 12).

[21] Tameem Adel, Taco Cohen, Matthan Caan, and Max Welling. «3D scattering transforms for disease classification in neuroimaging». In: *NeuroImage: Clinical* 14 (2017), pp. 506–517. DOI: 10.1016/j.nicl.2017.02.004. URL: http://eprints.gla.ac.uk/214136/ (cit. on p. 19).

[22] V. Chudácek, J. Andén, S. Mallat, P. Abry, and M. Doret. «Scattering Transform for Intrapartum Fetal Heart Rate Variability Fractal Analysis: A Case-Control Study». In: *IEEE Transactions on Biomedical Engineering* 61 (2014), pp. 1100–1108 (cit. on p. 19).

[23] Matthew Hirn, Stéphane Mallat, and Nicolas Poilvert. «Wavelet Scattering Regression of Quantum Chemical Energies». In: *Multiscale Modeling & Simulation* 15.2 (Jan. 2017), pp. 827–863. ISSN: 1540-3467. DOI: 10.1137/16m1075454. URL: http://dx.doi.org/10.1137/16M1075454 (cit. on p. 19).