Politecnico di Torino
Dipartmento di Scienza applicata e Tecnologia
MASTER DEGREE IN PHYSICS OF COMPLEX SYSTEMS

Connecting Structure and dynamics in model network glasses

Candidate:
Simone Monaco

Supervisors:
Dr. Frank Smallenburg

Prof. Alfredo Braunstein

Luglio 2020

Abstract

In recent literature, increasing experimental and numerical effort has been devoted to understanding and predicting the behavior of glassy systems. Among those classes of materials, of particular relevance are network glasses. In such system the composing particles can form a small number of bonds with their neighbors, leading then to an overall network structure. Among everyday materials, common window glass is perhaps the most obvious example of a network glass, but many others have, on the microscopic scale, a disordered network structure.

To better understand these materials, it is of primary importance to figure out the role and the local organization of the bonds in the network structure. In literature, a common way to study such systems is via the use of patchy colloids as a simplified model. In this thesis we analyze the behavior of patchy colloids by simulating them as *patchy particles*. These patchy particles consist of *hard-spheres* with a fixed number of patchy sites on their surfaces, through which the bonds can form. Starting from a simulated *2d* system of such particles, we compare its behavior with analytical results from Wertheim theory.

We then analyze the effect of the external conditions of the system by constructing an approximated phase diagram. Once the global behavior is fully characterized, the main goal of this thesis is to investigate how different local conditions drive the breaking and forming of bonds. In particular, we apply supervised learning techniques to predict which bonds are more likely to break in a certain configuration, looking only at the information given by the positions and connections of the particles. We essentially show which local parameters associated with each bond, from a *picture* of the system taken at equilibrium, are sufficient to train a neural network and infer the breaking time.

Even though the dynamics of the particles cannot be completely predicted by their position at a certain time, this approach can be a good starting point for understanding the interplay between local structure and dynamics in these complex materials.

Contents

List of Figures

List of Tables

List of Acronyms

HS Hard-Spheres 5

NN Neural Network 10

KF Kern-Frenkel 8

MC Monte Carlo 22

MD Molecular dynamics 22

EDMD Event driven molecular dynamics 22

LP Linear Perceptron 48

GD Gradient descent v

ML Machine Learning 44

MLP Multi-layer Perceptron 48

GN Graph Network 73

Colophon

This document was typeset using the XeTeX typesetting system created by the Non-Roman Script Initiative and the memoir class created by Peter Wilson. The body text is set 10pt with Adobe Caslon Pro. Other fonts include Envy Code R and Optima Regular. Most of the drawings are typeset using the TikZ/PGF packages by Till Tantau.

1.1 What are glasses

It is hard to overstate the importance of the role glasses have played in human history over the centuries. From art and architecture to engineering and modern technological devices, glasses have found a wide spectrum of applications in very different environments. But an important question with a not necessarily easy answer is: what is a *glass*?

In fact, even though many cultures in the history learned how to masterfully produce and treat glassy materials, even to this days this question remains poorly understood [3; 4; 5; 6]. From many studies, we know that when we rapidly cool down a liquid at a high temperature, it can reach a solid state before having enough time to organize itself in a crystal structure. When a liquid is cooled down beyond its freezing temperature T_m, but has not yet had time to crystallize, it forms a metastable state with respect to the crystal structure. We then get a *glass*. In a liquid we can define quantities such as *relaxation time* τ and *viscosity* μ, which are respectively the time the system needs to re-equilibrate after a perturbation and a coefficient expressing the resistance of the liquid to deformation. Both these two parameters can be used to quantify then the propensity of the system to flow

and change its conformation: the smaller they are, the faster the particles move. In a liquid at very high temperatures, τ is on the order of 0.1-1ps, and μ on the order of $10^{-4}\text{Pa} \cdot \text{s}$ [7]. Decreasing the temperature the system slows down, and these quantities increase more and more. When τ is high enough to be comparable with the experimental time-scales (around 100-1000s), the liquid would be too viscous to flow, behaving very similar to a solid. Then in literature this empirical point is usually known under the name of *glass transition temperature* (T_g) [3; 7], below which we can effectively assume to deal with a material identified as a glass. The fact that this transition originates from an empirical observation makes the definition of a glass somewhat ambiguous, and dependent on how long we are willing to continue our measurements. One other possible starting point is to classify a glass with respect to its differences with a crystalline solid. When particles in the system start to crystallize, they tend to form a geometrically ordered structure, where each of them occupies a fixed position with only small variations given by the thermal fluctuation. In glass formation, as already said, the system do not have the time to organize in such a way, even though the temperature is low enough to favor the crystal phase over the liquid one. The result is that we have a "liquid" system which would prefer to organize into a crystalline structure, but whose dynamics are so slow that it may take a practically infinite time to reach it. This means that the glass phase is an out-of-equilibrium state of the system. Both the inherent out-of-equilibrium nature, and the fact that the dynamics are extremely slow, make glassy systems extremely challenging to study.

1.1.1 Studying glasses

As a common approach in physics, the structure of a system can be reproduced and studied by the use of simplified models, enlightening only the most relevant features. The case of simple glasses is obviously not different. In past literature many different strategies have been applied to resolve the open questions surrounding the glass transition. There are two main ways of getting data on a glassy system: experiments and numerical simulations.

The most *natural* way to analyze a glass transition, which has also a long history in literature [8; 9; 10], is to take a molecu-

lar or atomic liquid and measure its behavior after *super-cooling*. This type of analysis reveals the most complex part of the glass transition problem: the fact that the relaxation time changes 14 orders of magnitude between the liquid and glass states. While experiments on atomic and molecular glasses in principle allow access to this wide range of time scales, they come with an important limitation: the small size of the building blocks forming the glass make it essentially impossible to study the local structure and dynamics of these systems in detail [6]. This restriction in fact limits the use of these experiments only to the observation of global properties.

One other possibility to explore the behavior of glasses is the use of colloids , which are suspensions of particles of one type of material in another substance. The interaction between the two kinds of particles in the mixture (particles and solvent) causes the suspended particles, which are generally bigger, to experience random forces from the smaller solvent molecules encountered during their motion. This phenomenon results in making these particle follow random path due to the fact that they are continuously deviated by the impacts, and was explained by Einstein in 1905 [11]. In order to have a stable colloidal suspension, we need particles that are significantly larger than a solvent molecule (generally $>$ 1nm) but small enough to move when impacted by solvent molecules (so smaller than a few micrometers). Although this random motion differs significantly from a ballistic motion usual of atomic systems on short time scales, colloidal particles still follow the same statistical physics of atoms and molecules. For this reason they are in the same way capable of forming gases, liquids, crystals and glasses. Then for our purposes a system of this kind will behave as an atomic or molecular one, with the difference that the particles (intended only as the bigger ones) forming it will move at a more treatable velocity.

From an experimental point of view, a colloidal dispersion can be obtained as a suspension of particles in any physical state in another substance potentially in a different state. A colloid then differs from a solution, in which only one phase is present, because it present a dispersed phase (the suspended particles) inside a continuous one. This means that, taking a cup of Cappuccino as a prototype colloidal system, we can see a liquid solution of milk and coffee, and a milk foam over it. The foam itself, differently

from the liquid below, is made of one liquid phase of milk with small clusters of air dispersed inside it. To be more precise, even the milk we have put inside our cup is actually a colloid, with discrete fat bubbles and protein molecules suspended into the liquid phase. Going over our Cappuccino, a colloidal dispersion can present a great variety of interesting features, mainly given by their structure based on the multiple length scales of the suspended particles and the molecules forming the solvent. First of all the dispersed particles generally have a larger size than single molecules, then they can more easily seen at the *single particle* level, fixing the drawback of atomic systems by providing direct access to their local structure. One other advantage of the larger size results in a slower dynamics, making their behavior easier to study in real time. Moreover, colloidal particles can be created out of a variety of materials, and tuned in terms of their size, shape, and surface chemistry. This allows us to adjust the interactions between them, making them an ideal playground for studying *e.g.* phase behavior, self-assembly, local structure, and glassy dynamics.

Computer simulations of such systems can be performed using e.g. Monte Carlo (MC) or molecular Dynamics (MD) algorithms. Respectively making the system "evolve" through the most probable configuration and integrating Newton's equations of motion for particles. The system is then described as a set of particles interacting with spherical symmetric potential (as the *Lennard-Jones* potential). Both classes of methods permit a more controllable evolution, where the particles position is known at every time, moving the *bottle-neck* of the problem only to the required computational power. This practically means that a simulation run for a finite amount of time is able to analyze only a small section in the time scale of the transition, namely a variation of about 5 decades of dynamic slowing of τ, making such simulations not really powerful in this task [6].

1.2 Patchy particles

One extremely versatile way of tuning the interactions between colloidal particles is by incorporating attractive sites on their surfaces. These sites impose a strict directionality on bonds

between particles, representative of what one might find in certain molecular systems.

The modeling of such potential requires some additional care, since a usual framework with particles under the effect of of an isotropic interaction, such as a *Lennard-Jones* potential, does not reflect those characteristics. One way commonly adopted in literature is the use of *patchy particles*, *Hard-Spheres* (HS) whose surface is decorated with attractive sites, which can interact to form bonds. These directional interactions can drastically impact the phase behavior and structure of the system, giving rises to the existence of substantially different equilibrium states, such as *empty liquids* [12] or *equilibrium gels* [13]. In particular, patchy particles with a fixed and small valence tend to reach a liquid phase only at low temperatures, organizing in long chains interconnected with each other in a small number of steps. This *quasi*-equilibrium state, characterized by peculiar properties, takes the name of *equilibrium gel* to emphasize the fact that the system reaches and get stopped in one of its multiple local minima, defined by the bonding pattern, but without incurring into changes of the gel mechanical and thermal properties. One other unusual characteristic that a patch particle model, with opportunely chosen patches features, can show is the presence of a re-entrant shape of the coexistence curve in the phase diagram, meaning that the density of the region in which the system is in the liquid phase decreases when the temperature is reduced [14]. The vast set of peculiar properties that can be observed in a patch-particle model by just modifying a small number of parameters reveals how this class of models can be extremely powerful in capturing the behavior of very different systems.

The first model made for patchy particles was introduced in the 1980s, analyzing molecular fluids [15]. The idea underlying this work was to develop a broader class of many-body systems. To this end, the model can be extended to many materials by changing different parameters, such as the position of the sites or their mutual interaction. A systematic definition of a patchy system will be given in the following chapters, while here we will focus on part of the history of this class of models. The common choice in all these works is that of modeling particles as hard (or soft) spheres carrying a spherical repulsive interaction due to excluded volume, plus a contribution associated with the *site*

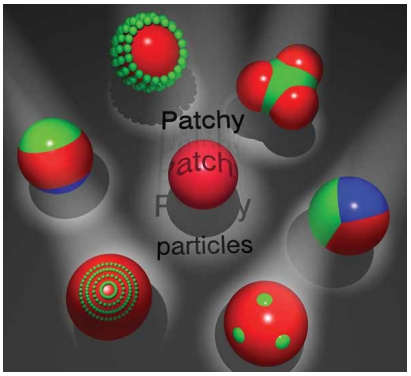


Figure 1.1: Computer representation of patchy particles with different types of sites positioned following predefined patterns. Illustration from [1].

- *site* interaction, often modeled as an approximate square-well potential. [16; 17].

As they are constructed, patchy particles have been used in literature to model any system that presents a strong association and a small coordination number of the particles. This feature is simply reflected by decorating the hard-spheres with a number of patchy sites that can mimic the system under investigation. Once the number and types of the sites has been defined, a proper interaction between them can be chosen. One other interesting patchy potential to look at was introduced by Nezbeda in 1989 [18], analyzing the thermodynamic properties of water. Each particles was reduced to a sphere with 4 bonds in a tetrahedral structure, two of them representing the hydrogen bonds and the other representing the oxygen ones. The different sites feel a square-well attraction in the proximity of sites of the opposite type. This means that a hydrogen site can form a bond only with a lone-pair site (the oxygen), and feels no interaction near another hydrogen site. The same reasoning applies for the oxygen sites. Nezbeda *et al.* demonstrated that just this simple model with short range interactions was sufficient to reproduce many of the anomalies of water and its phase separation processes [18]. In the same way, other groups applied similar models to describe other materials with network structures, such as silica

[19]. These works are the precursors of the models used in modern research projects on network glasses . What can be observed in silica, which actually is the material that forms window glasses, is the fact that the molecules can form a limited number of bonds between each others. This characteristic has the result that the overall structure organized itself effectively in a network topology, still having the previously underlined features of a *simple glass*. Network glasses can exist in both 2 and 3 dimensions, and the *dimensionality* is one important parameter affecting the property of the material. One other important parameter, partially related to dimensionality, is the *connectivity* of the network, which is the average number of bonding site each node in the network has. The combination of the two suggests some relations with the order of magnitude of the glass temperature transition, and is important in the formation of oriented fibers out of polymeric glasses [20]. Due to its capability of forming only 4 hydrogen bonds with each molecule, also (vitrified) water can be considered among the class of network glasses . In the same way *network-forming* liquid or glasses, which present a disordered structure but organized in a network topology, cover also amorphous silicon and polymers.

More recently, patchy particles are also attracting interest for the purpose of modeling biological systems, describing phenomena such as protein crystallization [21; 22]. The reason for that is again their ability to efficiently capture the short-ranged and anisotropic attraction between the proteins in solution. Even though proteins have a much more complex structure than water or silica molecules, it is possible to gain useful insights even with simple models. There are many ways to define the patchy sites and the pair potential, depending on which it is possible to capture different features of the real system. In 1999 Sear *et al.* introduced a minimal model in order to describe the coexistence of phases for globular proteins [21]. As in the primitive models previously described, the protein are described as HS decorated with a geometrical pattern of sites. Each site has now a numbered label and the attractive potential is such that odd sites interact only with even sites and vice versa. In this model the patches are defined as spots on the surface of the spheres, which can interact with similar spots of other spheres if all the conditions are fulfilled. Now the volume in which the short-range potential exists is considered as a physical extension of the particles. In this way the sites have a fixed height over

the surface and a bond forms only when two of them intersect. One similar approach, with more generality in the interaction between the differently labeled sites is the model introduced by Kern and Frenkel in 2003 [23]. This will be the one that we adopt for the modeling of our particles in the next chapters. With the *Kern-Frenkel* (KF) model is also possible to add features like having at most one bond at time for each site only by opportunely tuning geometrical properties of the particles. One other major advantage of the KF over models based on soft potentials is that the bonds are clearly defined, since each bond is either absent or fully formed, giving a fixed energy contribution.

The formation and organization of the bonds inevitably has a strong impact on different phases that can appear in the study of these systems [24; 25]. Besides the use of numerical simulations, the relation between the appearing phases can also be achieved by theoretical results. In particular it is possible to investigate phenomena like gas-liquid phase separation through the perturbation approach of Wertheim theory [16; 17]. This theory considers the bonds in a mean field approximation, interpreting their effect as mutually independent. Despite its intrinsic simplicity, this theory has been widely used to predict and understand the behavior of patchy particles . In the last few decades, many groups used its result to study thermodynamic properties of different systems [26; 27; 28].

Rather than simply study patchy colloidal particles through simulations, some groups also create in laboratory object with the same characteristics in different ways [29; 30]. The work of Biffi *et al.* uses DNA chains that can self-assemble to form *DNA stars*1.2. These stars can only form bond with the ending parts of each chain, which are constructed in such a way to make possible only bonds between *patchy sites* of the same type. This approach permits to create ideally any kind of patchy particles and has the advantage of of having biological objects that express all the relevant features at temperatures comparable to room temperature (the sequences self-assemble at $T_{\mathbf{sa}} \approx 65°C$) [30].

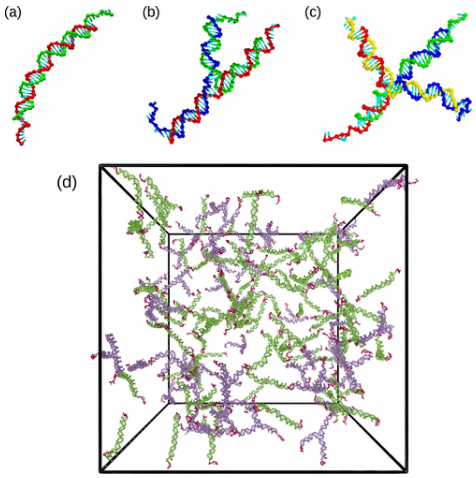


FIGURE 1.2: (a–c) Representations of DNA nanostars with different coordination number. (d) Simulation snapshot of a mixture of *dimers* and *trimers*. The image is taken from [2].

1.3 Aim of the thesis

Simple and network glasses materials, due their intrinsic complex behavior, are a fertile ground for the development of new simulation paradigms. Since the analytical study of these systems can provide us with only a limited range of results in predicting their behavior, new numerical approaches can instead lead us to notable improvements. In particular, many studies have proved how the characterization of the local network structure can be used to predict the dynamic of the system from both a macroscopic and a microscopic point of view [31; 32; 33]. This suggests the possibility of developing simulation algorithms to infer the topological changes, as formations and breaking of the bonds, starting from the structure itself. Thanks to its general applicability, the use of machine learning techniques is a natural choice to perform this task. The main idea, supported by many works from the last few years [34; 35], is to develop techniques able to catch the relevant features from the local structure and aggregation of the particle to determine the glassy dynamics, and to extend these promising methods also to network glasses .

By looking at a *movie* of the particles evolving in our network glass system, an observer could detect the evolution of the bonding pattern, then guess which configurations are more likely to break or to form. In this way, the observer could start considering the fact that some local conformations, which are generally more stable than others, may be stronger regardless of their history, and may conjecture the possibility convinced by the possibility that most of the future of the movie can be told just by looking at one single frame. This story is the key point of the thesis, in which we elaborate a way of translating the structural information contained in a single snapshot of a simulation to a set of features that can be used to feed a *Neural Network* (NN), which plays the role of the observer. Such NN can be *trained* through the observation of many snapshots of the system, then our goal is to see if it is effectively able to predict how the bonding pattern evolve.

Starting from a simulation framework of a 2-dimensional KF model, we extract some properties of the system, in order to characterize it and make a comparison with the analytical results that can be achieved. Then moving to the study of the construction and training of the NN, we investigate which information from the underlying system are more useful to define a dataset. In this first stage, the algorithm is implemented in order to predict the breaking time of a bond which are already formed in the snapshots contained in the dataset. In the same way a NN can be trained to predict the formation of bonds for each pair of particles, generating an instrument able to say which steps the system would follow from one snapshot to the other. Going further, an accurate algorithm predicting these information would then lead to an extremely fast simulation technique to reproduce the dynamics of model network glasses .

2.1 Description of the model

In this thesis we intend to analyze the behavior of a patchy colloid system, made of hard spheres (or disks) in a *2-dimensional* square volume. The model that will be described in this section is known under the name of KF model [23]. Each of the particles has a fixed diameter σ and a certain number M of short-ranged square-well attraction sites. This means that in addition to the non-overlapping potential of hard spheres each particle can interact with one other within a distance between σ and $\sigma_{\max}$, which is the range of this attracting potential, if the orientation of the two is such that 2 respective sites face each others. In this work all sites are assumed to be identical, so that each site can form bonds of the same strength with each of the other. Moreover, it will be enforced the fact that each site can form at most one bond at a time. This last assumption can be trivially achieved in numerical simulations of this kind of particles by opportunely choosing the radius of the disks and the aperture of the patchy sites.

More quantitatively each particle is modeled as a sphere (actually a disk in this *2-d* representation) of diameter σ, feeling all

the others through a non-overlapping potential of the form

$$U_{HS}(r) = \begin{cases} \infty & \text{if } r \leq \sigma \\ 0 & \text{if } r > \sigma \end{cases}. \tag{2.1}$$

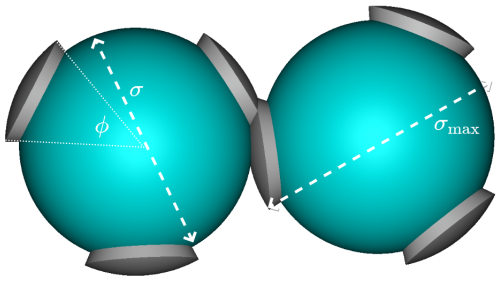


FIGURE 2.1: Illustration of two patchy particles, each consisting of a hard core with three equal attractive sites.

The bonding sites can then be graphically represented, as in Fig. 2.1, as M sections of an annulus of inner radius $\sigma/2$ ad outer radius $\sigma_{\max}/2$, equispaced along the circle, with a fixed aperture identified by $\delta := \cos\phi$. The total potential between two particles is then given by

$$U(\mathbf{i},\mathbf{j}) = U_{HS}(r_{ij}) + \sum_{\alpha\in\Gamma_i}\sum_{\beta\in\Gamma_j} U_W^{\alpha\beta}(\mathbf{r}_{ij},\mathbf{\Omega}_i,\mathbf{\Omega}_j), \tag{2.2}$$

with Γ_i denoting the set of M patchy sites of particle i, $\mathbf{i} = \{r_i;\ \Omega_i\}$ the shorthand for position and orientation, *and* $r_{ij} = ||r_i - r_j||$ the absolute distance between the two particles. $U_W^{\alpha\beta}(\mathbf{r}_{ij},\mathbf{\Omega}_i,\mathbf{\Omega}_j)$ is then the attractive well potential, dependent on $\mathbf{r}_{ij}$, $\mathbf{\Omega}_i$ and $\mathbf{\Omega}_j$. This is expressed as a square-well interaction as

$$U_W^{\alpha\beta}(\mathbf{r}_{\alpha\beta}) = \begin{cases} -\epsilon & \mathbf{C} \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The **C** condition fulfilled if and only if the sites are facing and they are close enough to interact. Naming $\hat{r}_{ij}$ the versor with

direction pointing from particle i to particle j and $\cos(\phi) = \delta$, this means that all these conditions have to be fulfilled together:

$$\mathbf{C} = \begin{cases} \hat{r}_{ij} \cdot \hat{d}_{i,\alpha} > \delta \\ -\hat{r}_{ij} \cdot \hat{d}_{j,\beta} > \delta \\ r_{ij} < \sigma_{\max} \end{cases} , \tag{2.4}$$

where $d_{i,\alpha}j$ and $d_{j,\beta}$ are respectively the versor pointing from the center of particle i to the center of patch α in particle i and the one pointing from the center of particle j to the center of patch β in particle j.

2.2 Wertheim theory

2.2.1 Description of the theory

One powerful way to predict the behavior of patchy particles and get some analytical results is Wertheim theory [16; 17]. The core idea of this approach is to perform a thermodynamic perturbation theory on a fluid of particles characterized by the possibility to form a limited number of bonds with each others. Starting from the unbonded system as the reference one, in a high temperature limit the effects of the bond is weak with respect to the others energetic terms and hence it is possible to consider it as a perturbation. We can then expand the partition function around that reference.

Neglecting the effect of the sticky sites the reference fluid can be expressed as a system of N particles interacting by a reference potential U_{ref}, for instance the one introduced to define hard spheres. Then defining $\rho = N/V$ as the density of the particles and $\mathbf{\Gamma}_i = \{\mathbf{r}_i, \mathbf{\Omega}_i\}$ as the set of positions and orientations for each particle, we can write down the partition function of the system as:

$$\mathcal{Z}_{\text{ref}} = \frac{1}{\Lambda^{2N} N!} \int \prod_i \mathrm{d}\mathbf{\Gamma}_i \; e^{-\beta \phi_{\text{ref}}(\mathbf{\Gamma}^N)} \tag{2.5}$$

and

$$\mathcal{F}_{\text{ref}}(N, V, T) = -\frac{1}{\beta} \log \mathcal{Z}_{\text{ref}}(N, V, T). \tag{2.6}$$

Here and in the following, β is defined as $\frac{1}{k_{\text{B}} T}$, then Λ the thermal wavelength and $\phi_{\text{ref}}(\mathbf{\Gamma}^N)$ the energy of the reference system related

to the given interaction potential. For an isotropically interacting reference system, $\mathcal{Z}_{\rm ref}$ depends only on the position of the particles since all the contribution of the orientation-dependent patchy sites will be collected in the perturbation term. In order to evaluate the contribution of this extra term it is then useful to consider the effect of the formation of one single bond. Without loss of generality, let us consider a bond between particles with label 1 and 2, obtaining a total potential energy of the form:

$$\phi_{tot}(\mathbf{\Gamma}^N) = \phi_{\rm ref}(\mathbf{\Gamma}^N) + \phi'(\mathbf{\Gamma}^N) = \phi_{\rm ref}(\{\mathbf{r}_i\}) + \phi'(\mathbf{\Gamma}_1, \mathbf{\Gamma}_2), \quad (2.7)$$

in which the extra term ϕ' is the perturbation due to the formation of this bond. In this framework the orientation of the particles is only relevant for the particles making the bond and ϕ' will be negative when the particles 1 and 2 are positioned in such a way to make possible the formation of the bond (*i.e.* in our model following the condition in Eq. 2.4). In the following the integrals without subscription will denote as before an integration on all possible positions and orientations, and an integral with the subscription *bond* will denote an integration only made over positions and orientations satisfying the condition in Eq. 2.4.

We can then write down the partition function of the system with a single bond, given by

$$\begin{aligned}\mathcal{Z}_{bond} &= \frac{1}{\Lambda^{3N} N!} \int_{bond} \prod_{k\neq 1,2} \mathrm{d}\mathbf{\Gamma}_k \; \mathrm{d}\mathbf{\Gamma}_1 \mathrm{d}\mathbf{\Gamma}_2 \; e^{-\beta\phi_{\rm ref}(\mathbf{\Gamma}^N)} e^{-\beta\phi'(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2)} \\ &= \frac{1}{\Lambda^{3N} N!} \int \prod_{k\neq 1,2} \mathrm{d}\mathbf{\Gamma}_k \int_{bond} \mathrm{d}\mathbf{\Gamma}_1 \mathrm{d}\mathbf{\Gamma}_2 \; e^{-\beta\phi_{\rm ref}(\mathbf{\Gamma}^N)} e^{-\beta\phi'(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2)}.\end{aligned} \quad (2.8)$$

Then after defining the pair correlation function $\rho^{(2)}_{\rm ref}(\mathbf{\Gamma}_i, \mathbf{\Gamma}_j)$ of the reference system as

$$\begin{aligned}\rho^{(2)}_{\rm ref}(\mathbf{\Gamma}_i, \mathbf{\Gamma}_j) &= N(N-1) \frac{\int \prod_{k\neq i,j} \mathrm{d}\mathbf{\Gamma}_k \; e^{-\beta\phi_{\rm ref}(\mathbf{\Gamma}^N)}}{\int \prod_k \mathrm{d}\mathbf{\Gamma}_k \; \mathrm{d}e^{-\beta\phi_{\rm ref}(\mathbf{\Gamma}^N)}} \\ &\propto N(N-1) \frac{\int \prod_{k\neq i,j} \mathrm{d}\mathbf{\Gamma}_k \; e^{-\beta\phi_{\rm ref}(\mathbf{\Gamma}^N)}}{\mathcal{Z}_{\rm ref}},\end{aligned} \quad (2.9)$$

we can write

$$\begin{aligned}\frac{\mathcal{Z}_{bond}}{\mathcal{Z}_{\text{ref}}} &= \frac{1}{\mathcal{Z}_{\text{ref}}} \int \prod_{k\neq 1,2} \mathrm{d}\mathbf{\Gamma}_k \int_{bond} \mathrm{d}\mathbf{\Gamma}_1 \mathrm{d}\mathbf{\Gamma}_2 \; e^{-\beta\phi_{\text{ref}}(\mathbf{\Gamma}^N)} e^{-\beta\phi'(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2)} \\ &= \frac{1}{N(N-1)} \int_{bond} \mathrm{d}\mathbf{\Gamma}_1 \mathrm{d}\mathbf{\Gamma}_2 \; \rho^{(2)}_{\text{ref}}(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2) e^{-\beta\phi'(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2)}.\end{aligned} \tag{2.10}$$

The pair correlation function expresses the probability density associated with simultaneous finding two particles at two specific positions and orientations, integrating out the effect of all the others. Therefore, the presence of symmetries in the system can simplify its structure a lot. In particular, since the reference fluid is isotropic and homogeneous, $\rho^{(2)}_{\text{ref}}$ is only dependent on the absolute distance between the 2 particles. In this case, the pair correlation function is directly related to the radial distribution function g_{ref}, characteristic of the system such that

$$\rho^{(2)}_{\text{ref}}(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2) = \rho^2 g_{\text{ref}}(|\mathbf{r}_1 - \mathbf{r}_2|) = \rho^2 g_{\text{ref}}(r_{12}). \tag{2.11}$$

Such a function takes the name of radial distribution function. Substituting this into Eq. 2.10 we obtain

$$\begin{aligned}\frac{\mathcal{Z}_{bond}}{\mathcal{Z}_{\text{ref}}} &= \frac{\rho^2}{N(N-1)} \int_{bond} \mathrm{d}\mathbf{\Gamma}_1 \mathrm{d}\mathbf{\Gamma}_2 \; g_{\text{ref}}(r_{12}) e^{-\beta\phi'(\mathbf{\Gamma}_1,\mathbf{\Gamma}_2)} \\ &\simeq \frac{1}{V} \int_{bond} \mathrm{d}\mathbf{\Gamma}_{12} \; g_{\text{ref}}(r_{12}) e^{-\beta\phi'(\mathbf{\Gamma}_{12})},\end{aligned} \tag{2.12}$$

where the last line is obtained by integrating out one of the dummy variables and and taking the large N limit.

The evaluated ratio can be associated with the first correction on the free energy of the full system, related to the formation of a single bond. Such extra contribution is the difference between the free energy of the bonded system and the reference one, so

$$\begin{aligned}\Delta\mathcal{F} = \mathcal{F}_{bond} - \mathcal{F}_{\text{ref}} &= -k_b T(\log \mathcal{Z}_{bond} - \log \mathcal{Z}_{\text{ref}}) = -k_b T \log \frac{\mathcal{Z}_{bond}}{\mathcal{Z}_{\text{ref}}} \\ &\simeq -k_b T \log \left(\frac{1}{V} \int_{bond} \mathrm{d}\mathbf{\Gamma}_{12} \; g_{\text{ref}}(r_{12}) e^{-\beta\phi'(\mathbf{\Gamma}_{12})} \right).\end{aligned} \tag{2.13}$$

At this point, it is important to emphasize the fact that all the previous reasoning is fully general and therefore valid for every possible model provided that the reference fluid is isotropic and homogeneous. In order to proceed with the calculation it is then necessary to make some further assumptions. In particular, we need expressions for g_{ref} and ϕ'. One of the simplest possible choices for ϕ' is the one from the KF model described in 2.3, and so in this framework the integral performed onto the *bond*-subspace are simply performed onto the space fulfilling the condition 2.4. Then, since we are performing an expansion for a small effect of the interaction, we can just start by neglecting it in g_{ref}. This can be done by imposing $g_{\text{ref}}(r) = 1$, which means that the particles do not interact with each other (even neglecting the excluded volume interaction) and is a reasonable approximation for a dilute gas ($\rho \simeq 0$). Under these extra assumptions Eq. 2.13 becomes:

$$\begin{aligned}\Delta\mathcal{F} &= -k_b T \log\left(\frac{1}{V}\int_{bond} \mathrm{d}\Gamma_{12} e^{\beta\epsilon}\right) \\ &= -\epsilon - k_b T \log\left(\frac{1}{V}\int_{bond} \mathrm{d}\Gamma_{12}\right) \\ &= -\epsilon - k_b T \log\left(\frac{V_{bond}}{V}\right).\end{aligned} \tag{2.14}$$

Here V_{bond} is defined as the bonding volume fulfilling the condition of Eq. 2.4. In this formulation the two contributions to the change in free energy are an energetic term, associated with the energy gain due to the formation of a bond, and an entropic one, which expresses the loss of entropy due to the constraint imposed by the fact that one of the bonded particles has to stay in the bonding region of the other.

Once the effect of the formation of a single bond on the free energy of the system is clear, it is then possible to extend this approach to any number of bonds. In the KF model each particle has M patches to form bonds with, and so the full system has MN patches in total. The contribution of all the possible bonds will be performed in a combinatorial way by treating them as before as a perturbation on the reference system. First of all, the free energy of the reference system can be considered to have the same structure as the one of the full model, but with a contribution

per bond equal to 0. In this way even in the reference system there will exists an extra quantity, n_b^{ref}, expressing the number of "accidental" bonds, in the sense that even if they are present, their presence does not affect the free energy count. If one assumes that for each patch the average number of bonds is given by the integration over the bonding space between two particles (again without loss of generality taken as 1 and 2), the total number of bonds (avoiding double counting) is given by:

$$n_b^{ref} = \frac{NM^2}{2} \int_{bond} \mathrm{d}\mathbf{\Gamma}_{12}\, g_{\mathrm{ref}}(r_{12}). \tag{2.15}$$

We now want to consider the impact of adding multiple additional bonds on the free energy. However, since each added bond has a small impact on the overall structure of the system, it is not trivially true that each successive bond changes the free energy by the same amount. The simplest assumption that can be done to get rid of the mutual relation between the formed bonds is to adopt a mean-field approach, and so to consider the free energy cost of the creation of one as the same for all of them. Such an assumption can be seen as a mean-field approximation because it neglects any interactions between the bonds. It also implies that the structure of the fluid, and therefore the function $g(r)$, does not change much with the addition of the bonds. Moreover we will have to neglect the formation of closed loops, or at least assume that they have no extra contribution to the free-energy.

To see how this approach treats ring formation, consider the fact that in an exact framework it would be reasonable to add separately the contribution of each new bond as it is added to the system as long as the involved particles do not come from the same cluster. However, when adding a bond between two particles already connected by a path of existing bonds, the addition of an extra one would modify the free-energy contribution of all the others in this *now closed* path, invalidating the assumption that the effect of all bonds is uncorrelated. Moreover the presence of the ring would add more constraints to all the particles of the sequence, causing a possibly non-negligible loss of entropy. Although there is no simple way to evaluate the effect of the extra bond forming a ring with respect to the energy contribution of a simple string with the same number of bonds,in Wertheim approximation this problem simplify assuming those configurations as equivalent,

saying that the effect of forming the extra bond to close the ring is the same as the effect of forming any other bond. This assumption is more reasonable when the number of rings is small and so the error committed in ignoring them is not so relevant.

Both these considerations are reasonable if and only if the total number of bonds is low, which is equivalent to having the system in a high temperature regime. In this framework the total free energy for n_b formed bonds is:

$$\mathcal{F} = \mathcal{F}^{ref} + \left(n_b - n_b^{ref}\right) \Delta\mathcal{F} + \mathcal{F}_c(n_b) - \mathcal{F}_c(n_b^{ref}). \tag{2.16}$$

Where $F_c(n_b)$ is the combinatorial free energy contribution associated to the degenerate number of possible associations of particles with the same n_b, which can be written as:

$$\mathcal{F}_c(n_b) = -k_b T \log\left[\frac{[(MN)(MN-1)]\cdot[(MN-2)(MN-3)]\ldots[(MN-2n_b+2)(MN-2n_b+1)]}{n_b! 2^{n_b}}\right]. \tag{2.17}$$

In this equation, each product in square parenthesis inside the logarithm represent a bond, which can be formed between all the remaining patchy sites not already involved in the previous ones. It is easy to see that the numerator has n of these products. Then to avoid double-counting the total number is divided among all the possible permutations of the n bonds and the two sites in each of them. Then by defining $p_b = \frac{2n_b}{MN}$ as the probability of a patch to be part of a bond and using Stirling's approximation Eq. 2.17 becomes:

$$\begin{aligned}\beta\mathcal{F}_c(n_b) &= -\log \frac{(MN)!}{(MN-2n_b)!\, n_b!\, 2^{n_b}} \\ &= \frac{MN}{2}\left[p_b + 2\log(1-p_b) + p_b \log \frac{p_b}{MN(1-p_b)^2}\right].\end{aligned} \tag{2.18}$$

At this point it is important to note that Stirling's approximation applied to the previous equation is valid for n_b and MN large, and $MN - 2n_b \gg 1$. Considering the relation between n_b and MN, these conditions are all valid in the thermodynamic limit of large systems ($N \to \infty$), as long as the fraction of bonded sites is finite. Then coming back to Eq. 2.16 and minimizing $\mathcal{F}$ with

respect to p_b in order to get the optimal value we get:

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial p_b} &= \frac{\partial n_b}{\partial p_b}\Delta\mathcal{F} + \frac{\partial \mathcal{F}_c(p_b)}{\partial p_b} \\ &= \frac{MN}{2}\Delta\mathcal{F} + \frac{1}{\beta}\frac{MN}{2}\log\frac{p_b}{MN(1-p_b)^2} = 0,\end{aligned} \tag{2.19}$$

where $p_b^{ref} = \frac{2n_b^{ref}}{MN}$. Solving this for p_b and repeating for $\frac{\partial \mathcal{F}}{\partial p_b^{ref}}$, we obtain then

$$\frac{p_b}{MN(1-p_b)^2} = e^{-\beta\Delta\mathcal{F}}\ ; \quad \frac{p_b^{ref}}{MN(1-p_b^{ref})^2} = e^{-\beta\Delta\mathcal{F}}. \tag{2.20}$$

As previously said, in the second relation of Eq. 2.20, $\Delta\mathcal{F}$ is calculated in the absence of patchy interaction ($\epsilon = 0$). This minimization give us an expression for $\Delta\mathcal{F}$ as a function of the probability of formation of a bond, plugging it into Eq. 2.16 and using Eq. 2.18 for the combinatorial free energies we get:

$$\begin{aligned}&\mathcal{F}_c(p_b) - \mathcal{F}_c(p_b^{ref}) \\ &= \frac{MN}{2}\left[(p_b - p_b^{ref}) + 2\log\frac{1-p_b}{1-p_b^{ref}} - \beta(p_b - p_b^{ref})\right]\end{aligned} \tag{2.21}$$

and so

$$\begin{aligned}\beta f := \frac{\beta\mathcal{F}}{N} &= \frac{\beta\mathcal{F}_{\text{ref}}}{N} + \frac{M}{2}(p_b - p_b^{ref}) + M\log\frac{1-p_b}{1-p_b^{ref}} \\ &\simeq \frac{\beta\mathcal{F}_{\text{ref}}}{N} + \frac{M}{2}p_b + M\log(1-p_b)\end{aligned} \tag{2.22}$$

Where the second line can be obtained by observing that in the low density regime we are looking at it is reasonable to assume $p_b^{ref} \simeq 0$.

2.2.2 Predicting the number of bonds

The first relation of Eq. 2.20 can be solved in order to get the probability of formed bonds in the system. This result [36], which can be figured out as a function of the density and of the temperature will be a good starting point to test the simulation

framework. Multiplying both sides by V and defining $x = 1 - p_b$ the fraction of non-bonded sites, this equation becomes:

$$\begin{aligned} 1 - x &= M\rho\Delta x^2, \\ x &= \frac{2}{1 + \sqrt{1 + 4M\rho\Delta}}, \end{aligned} \tag{2.23}$$

with

$$\Delta := Ve^{-\beta\Delta\mathcal{F}} = \int_{bond} \mathrm{d}\mathbf{\Gamma}_{12}\; g_{\mathrm{ref}}(r_{12}) e^{-\beta\phi'(\mathbf{\Gamma}_{12})}. \tag{2.24}$$

The last relation for the new quantity Δ directly derive from Eq. 2.13.

As discussed in the previous section, one first guess for evaluating $g_{ref}(r)$ can be made by observing that when the interaction between particles is negligible (*i.e.* in the ideal gas limit), the pair correlation of Eq. 2.11 does not depend on the distance between the particles and so is simply ρ. This limit can be achieved at low density, in which each particles is reasonably far from all the others, then $g_{\mathrm{ref}}(r) \simeq 1$. A comparison for this approximation has a very poor agreement with the with the numerical results, making necessary to look for a more accurate analysis. One improved expression for the radial distribution can be obtained by calculating it through its relation with the *compressibility factor* Z, defined as:

$$Z = \frac{p}{\rho k_b T}. \tag{2.25}$$

This parameter is useful to measure how much the current system is far from an ideal gas, for which Z would be exactly equal to 1. Thanks to the virial expression [37], this quantity can be written as a function of ρ, or better of its the dimensionless form $\eta := \rho\pi(\sigma/2)^2$, defined as the packing fraction. For small density ρ (*i. e.* for small η), Z can be expressed as a power expansion as follows:

$$\begin{aligned} Z(\eta) &= 1 + B_1\eta + B_2\eta^2 + \cdots \\ &= 1 + 2\eta \sum_{i,j} f_i f_j g_{\mathrm{ref}}(\sigma). \end{aligned} \tag{2.26}$$

The previous relation, for a 2 dimensional system under the assumption, valid in this treatment, that all the particles have the

same diameter. In the last line the factors f_i represent fraction of particles of all the possible species present in the system, defined as the mole fraction ρ_i/ρ. Since the analyzed system has only one type of particle, the relation can be simplified as:

$$Z = 1 + 2\eta g_{\text{ref}}(\sigma), \tag{2.27}$$

from which:

$$g_{\text{ref}}(\sigma) = \frac{Z-1}{2\eta}. \tag{2.28}$$

Moreover there exist many relations that can approximate the behavior of Z [38]. One of the simplest ones is known under the name of *Scaled particle theory*, and states:

$$Z(\eta) = \frac{1}{(1-\eta)^2}. \tag{2.29}$$

Now, recalling than $\phi'(\mathbf{\Gamma}_{12})$ can be expressed through the KF model as in Eq. 2.3, the equation for Δ can be evaluated. Considering that the only dependence on the orientation is contained in the Boltzmann weight, Eq. 2.24 become:

$$\begin{aligned}\Delta &= \int_{bond} \mathrm{d}\mathbf{r}_{12}\; g_{\text{ref}}(r_{12}) \langle e^{-\beta\phi'(\mathbf{\Gamma}_{12})} \rangle \\ &= e^{\beta\epsilon} \int_{bond} \mathrm{d}\mathbf{r}_{12} \frac{\mathrm{d}\Omega_1 \mathrm{d}\Omega_2}{(2\pi)^2}\; g_{\text{ref}}(r_{12}).\end{aligned} \tag{2.30}$$

Since the bonding condition makes the integral different from zero only in the space fulfilling condition 2.4, the integral over r_{12} can be estimated by assuming that $g_{\text{ref}}(r)$ is almost constant while going from radius σ to $\sigma_{\max}$, and so we can substitute it with its value in the middle of the range. The angular integral can be instead calculated considering that the two vector connecting the particles involved in a bond with the center of mass of both can span an angle of $\phi = \cos^{-1}\delta$, giving the overall result:

$$\begin{aligned}\Delta &\approx \frac{e^{\beta\epsilon}}{2\pi}(2\phi)^2 g\left(\frac{\sigma_{\max}-\sigma}{2}\right)\int drr \\ &\approx \frac{e^{\beta\epsilon}}{\pi}\phi^2 \frac{2-\eta}{2(1-\eta)^2}(\sigma_{\max}^2 - \sigma^2)\end{aligned} \tag{2.31}$$

$$\rho\Delta(\rho\sigma^2) = \frac{e^{\beta\epsilon}}{\pi}\phi^2 \frac{16 - 2\pi\,\rho\sigma^2}{(4-\pi\,\rho\sigma^2)^2}(\chi^2-1)\rho\sigma^2.$$

The last line of the previous equation emphasize the dependence on the dimensionless quantity $\rho\sigma^2$ of $\rho\Delta$, contained in Eq. 2.23, which can be proved to be the absolute probability to form a bond.

2.3 Simulation methodology

One other solution to the impossibility of exactly treating an N-body system can be provided by the use of numerical simulation. There are substantially two main strategies to perform this task, which are *Monte Carlo* (MC) and *Molecular dynamics* (MD) simulations. The first one solves the challenge by making the system evolve through proposing random displacements of single particles or portions of the system, drawing them from probabilistic distributions that are related to the system conditions. In this way each snapshot of the system taken after one movement is one possible configuration at the given condition, but looking at the full evolution we will not see the trajectory a real system would follow in the phase state. One other possible way of thinking, which has the advantage of adhering to the *true* evolution of the system, is given by MD. In particular, one common choice in simulating HS models, which we used in this work, is known under the name of *Event driven molecular dynamics* (EDMD) [39]. While in a usual MD approach the evolution of the system is performed by discretizing the time into steps of duration Δt and integrating the equation of motion at each step to update the position and momenta of the particles, in EDMD the evolution of the system is resolved analytically until an event (*i.e.* a collision between the particles) occurs. This idea can be performed by constructing an ordered list of all the events that may take place based on the current particle trajectory, which neglecting the potential interactions are just straight lines, and updating the list after each event.

Describing our particles as HS (which in our $2d$ case are just disks) of diameter σ, each event to collect is the situation in which two trajectories lead to a sufficiently small distance between the their associated disks. To also consider the effects of interparticle interactions, we start by also including a simple symmetric square-well potential, representing the possibility that the disks can bond

within a certain distance but not enforcing the directionality of the bonds, namely:

$$U_{\text{symm}}(r) = \begin{cases} \infty & \text{if } r \leq \sigma_H \\ \epsilon & \text{if } \sigma_H < r \leq \sigma_w \\ 0 & \text{if } r > \sigma_w \end{cases} . \tag{2.32}$$

This reasoning can be extended to the patchy particles we have studied in the previous sections by enforcing the fact that a bond can form only in the direction of the sites, *i.e.* by incorporating in the code the condition of Eq. 2.4. From the previous notation we would impose $\sigma_H \equiv \sigma$ and $\sigma_w \equiv \sigma_{\max}$. One more precise look into this will be given at the end of the section.

Independent of the the way bonds are formed, we can substantially have two classes of interaction: the one in which the hard cores of involved particles collide but they still remain bonded, or the one in which they enter or leave their mutual potential well, changing their bonding relation. In the first case we expect from the laws of dynamics an elastic impact, which conserves both kinetic energy and momentum. In the second case we have only momentum conservation, while the kinetic energy is expected to change of on amount equal to ϵ due to the *total* energy conservation. One specific interaction between two identical particles i and j, separated by a distance $\mathbf{r} = \mathbf{r}_i - \mathbf{r}_j$ and with relative velocity $\mathbf{v} = \mathbf{v}_i - \mathbf{v}_j$, will collide after a time τ which is the smaller positive solution (if it exists) of the relation

$$\mid \mathbf{r} + \mathbf{v}\tau \mid = \sigma. \tag{2.33}$$

Of course we can expect an impact of both classes by putting σ_H or σ_w in place of σ. Then in the same way for both the diameters the solution will be of the form

$$\tau = \frac{-b + s\sqrt{b^2 - v^2(r^2 - \sigma^2)}}{v^2} \qquad \text{with } \; b = \mathbf{v} \cdot \mathbf{r} \tag{2.34}$$

The value of s, which can take values of $+1$ and -1, is set to give the smaller positive solution of the equation and depends on the situation in which the particles are before and after the collision. If we start with two bonded particles (therefore $\sigma_H < r < \sigma_w$),

we can have a core collision, then $\sigma = \sigma_H$ and $s = 1$, and a well collision. Such collision can result in either a bounce within a well or a breaking of the bond, and in both cases we will have $\sigma = \sigma_w$ and $s = 1$. The difference between the two depends on the starting kinetic energy, meaning that the bond breaks if the relative motion between the two particles is strong enough to overcome the energy barrier of height ϵ.The last possible impact is finally a superposition of the wells, originating a bond. In this situation we will have $\sigma = \sigma_w$ and $s = -1$.

Imposing then the conservation of the momenta, we expect that the change in the velocity of a particle would be the same with opposite sign of the velocity change of the other, namely:

$$\delta \mathbf{v}_i = -\delta \mathbf{v}_j := \phi \mathbf{r}. \tag{2.35}$$

This variation ϕ can be calculated by imposing energy conservation. We can introduce the effect of the eventual bonding/ unbonding event as the variation δu, which can be either $0, +\epsilon, -\epsilon$, depending on the kind of impact we have. With m the mass of a particle, will then have:

$$\begin{aligned} \frac{m}{2}\left(\mathbf{v}_i^2 + \mathbf{v}_i^2\right) &= \frac{m}{2}\left(\mathbf{v}_i^2 + \mathbf{v}_i^2 + 2\phi^2, r^2 + 2\mathbf{v}\cdot\phi\mathbf{r}\right) + \delta u \\ &\implies \frac{\delta u}{m} + \phi^2 r^2 + b\phi = 0, \end{aligned} \tag{2.36}$$

having

$$\phi = \frac{-b + s\sqrt{b^2 - 4r^2\delta u/m}}{2r^2} \tag{2.37}$$

For a direct collisions between cores, $\delta u = 0$ and $r = \sigma_H$, while a collision between wells can be found with $\delta u = \pm eps$ and $r = \sigma_w$. We can find the energetic condition for the breaking of the bond by looking at at the case of the particles reaching a distance of σ_w from inside. In fact, in this case the only possible solution when $b^2 < 4\sigma_w^2\epsilon/m$ is obtained by imposing $\delta u = 0$. Then we would have a bounce event ($\delta u = 0$), with solution $\phi = -b/\sigma_w^2$.

In this way it is possible to give a complete physical analysis of a single event in the system. Naively, we can simply predict such events for all pairs of particles in the system. However, as the system size increases, this leads to an extremely large number of possible collision events, resulting in memory issues concerning

the storage of the event list. A solution to this problem and some other practical implementation concerns are briefly discussed in the following Section.

2.3.1 Implementation

One first complication that may come from the simulation of HS lies in the assignment of initial particle positions. In a usual MD simulation the common choice would be the one of setting a random particles configuration. The same can be applied to HS, but we have to consider the fact that no disk is allowed to fall in the area occupied by the others. This condition can be fulfilled by randomly inserting only one particle at time, checking each time if the new configuration has no overlap, and repeating the attempt if an overlap was created. This approach is obviously slower than a single random assignment, and becomes problematic for high-density simulations, in which each particle assignment from a certain time on can take an arbitrary long time to find an admitted configuration, leading to a potential infinite time even for the initialization process. One possible solution to this problem is given by assigning a fixed position to each particle, *e. g.* the points of a square grind, in order to have a pre-defined valid set of position. Making the system evolve for a certain time, it will typically recover a random configuration, as the one we would get after equilibrating it generated with the previous assignment strategy.

At the end of the initialization, each particle has a position and a velocity, which define the trajectory the particle will follow. All the intersection in the future between these trajectories can be collected in the time-ordered sequence of collision. Since the route of a particle get modified after any event involving it, only the first event in the system is guaranteed to take place, as other events may be recalculated after a collision. Once the occurrence τ of the first event has been found, the whole system can be allowed to evolve updating all the particle position up to that time. After a collision, the only features that can change are the velocity of just the particles involved in this collision, accordingly to Eq. 2.37, and eventually the bonding relations between them. Since each event involves a only two particles, we only have to check for new events to add for them. This strategy is in principle valid, but

generates one practical issue: in a system of N particles, each of them can collide with all the $N-1$ others. This means that the computational cost required to find the possible collisions for each particle involved in an event scales as the system size, making each update of the list considerably computationally heavy. One way to overcome this problem is given by separating the volume into smaller cells, then consider for each particles the collision with just the particles in the same cell or in the adjacent ones. The cell size is typically chosen as a multiple of the disk diameter, in order to fix the maximum number of particles each cell can contain to a value independent of N. In order to use this strategy it will be then necessary to add the fact that one particle crosses the cell border as a collected event. In the same way we can treat also periodic boundary conditions, by simply imposing that a particle arrives to the cell in the opposite size of the system when passing through a border. This extra kind of events to consider forces the system to stop an overall greater number of times, but each of them with a lower computational cost if the cell size has been properly chosen. The way of storing and optimally dealing with this list is then crucial to have quicker simulations and makes it possible to work with bigger systems [40].

The use of EDMD can be extended to the anisotropic patchy particles model we introduced in the previous sections by using the different bonding conditions and predicting the behavior of the particles once a bond is formed or broken. Both tasks are much less straightforward than the isotropic case, in fact they require the use of numerical algorithms [41; 42]. What we can analytically predict from Eq. 2.34 are the times two incoming particles would have spent to reach their well and core surfaces. in particular we expect that the collision time τ should stay between the time spent by the wells to touch each others (denoted as $t_{\min}$) and the time spent by the hard cores to collide ($t_{\max}$). Obviously if the calculation for $t_{\min}$ gives no solution,*i. e.* if the two particle never come close enough to interact, there is no need to continue. If the condition for finding $t_{\min}$ is already fulfilled ($\mathbf{r} < \sigma_w$) when the algorithm start looking for it, we can set $t_{\min} = t_{\text{current}}$. While if the hard cores never impact, $t_{\max}$ can be set as the first time at which r exceeds σ_w again. This provides us a limited time window to scan for the possible time τ at which a patch collision may occur. Then we introduce the functions $\Phi_i = \Phi_i(\mathbf{r}_{ij}, \{p_i\})$

and $\Phi_j = \Phi_j(\mathbf{r}_{ji}, \{p_j\})$, where $\mathbf{r}_{ji} = -\mathbf{r}_{ij}$ are the distance vectors between the two particles and $\{p_i\}$ and $\{p_j\}$ are the sets of the patches orientation, which express how close the patches are to interact. They can be written as:

$$\Phi(\mathbf{r}, \{p\}) = \max_{p \in \{p\}} \left(\hat{r} \cdot \hat{p} - \cos(\theta_m^p)\right). \tag{2.38}$$

If Φ_i is negative, none of the patches of particle i are oriented in the direction of $\mathbf{r}$. This means that we can only have a bond formation when $\Phi_i \cdot \Phi_j > 0$. By looking at the evolution of these two functions in time is then possible to determine the value of τ numerically. In particular we have a bonding event when $\Phi_i(t) = 0$ when $\Phi_j(t)$ is already positive. A bond breaking can instead occur (still doing energetic considerations as before) at the first time in which both $\Phi_i(t)$ and $\Phi_j(t)$ become 0. After the well event time has been numerically found, it can be added to the event list.

2.3.2 Comparison of theoretical and numerical results

At the end of this analytical treatment, we were able to express the fraction of bonded particles as a function of the dimensionless density by the use of Eq. 2.23. The same quantity can be obtained as average of many samples taken after the equilibration of the simulated system. Then, in order to enforce the fact that the algorithm actually simulates the physical system described by Wertheim theory , we can compare these results. This is the first step that will permit us to go beyond the analytic treatment we have done at this point, then to explore the features of the real system the theory is not able to predict. In the following are present some comparisons for different temperatures.

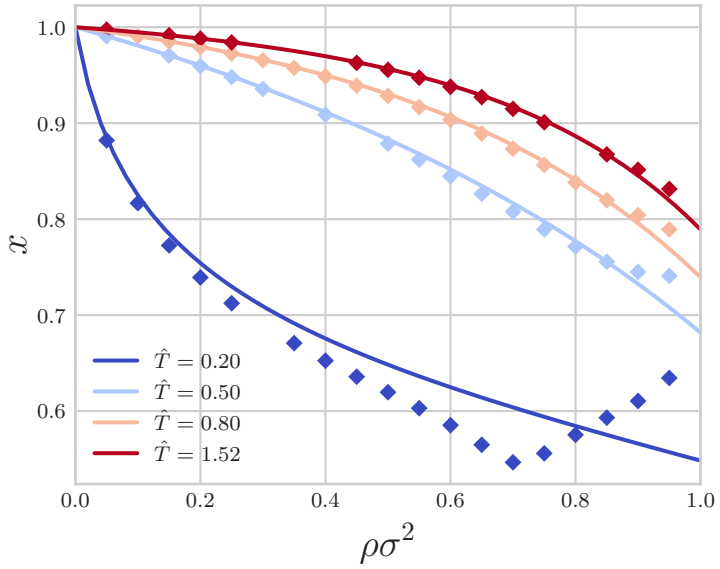


Figure 2.2: Plot of the comparison between the fraction x of unbonded patches evaluated via Wertheim theory (lines) and simulations (points), for different temperatures. Here the adimensional temperature is $\hat{T} = (\beta\epsilon)^{-1}$

In Fig. 2.2 are presented the analytical and numerical results, respectively as solid lines and squares of the same colours representing the points of the measures. As can be seen the two curves are more and more similar as the temperature increase, meaning that the analytical approximations are actually more accurate in those situations. This result is in total accordance with the previously stated assumptions.

2.4 Characterization of the Phase Behavior

As seen in the previous Section, the thermodynamic conditions of the system are crucial in order to understand its overall connectivity. From basic statistical physics, it is easy to understand that bonds between particles are more likely to form at low temperature and high density. A simple reason for this result is given by the fact that at low temperatures, bonds are harder to break due to lower particle velocities, and at high densities, the probability that particles find each other during the random movements obviously increase. What can be noticed so far is that even if we are able to fully understand global quantities, such as the concentration of the bonds, Wertheim theory does not predict how these bonds are organized. Systems of patchy particles , like many other materials, can exist in a variety of phases (e.g. gas, liquid, crystal), and one important step in studying them is to map out where we should expect these phases. Moreover, one of the reasons a direct application of Wertheim theory does not work well for predicting bond concentrations at low temperatures is the fact that you have not taken into account the possibility of phase coexistences yet. Nevertheless, Wertheim theory can also be used to predict phase coexistence [12].

As we decrease the temperature, we can expect to see the formation of clusters of particles of different sizes, related in some way with the system conditions. Since formation, distribution and size of such clusters are in fact what determines the physical state of the system, a deeper comprehension in this direction is crucial for the determination of its the properties. A further step can be therefore achieved by analyzing the phase behavior of the model, in order to look for the presence of regions in which each admitted state can be reached. Starting from various theoretical analysis [27], we can expect to see both fluid and crystal phases, but also phase separated regions and gelation. Knowing the phase our simulation is in is important because many quantities, such as the diffusion coefficient and the local structure of the system, are strongly related to the phase behavior. This implies that, in order to correctly interpret our results, we have to be certain of the phase we are simulating. In particular, in most cases, we want to ensure that our simulations contain a single phase, rather than a coexistence between two phases. A direct phase coexistence in

fact can lead to strong non-equilibrium and finite-size effects due to the interface that forms between the two phases.

Statistical mechanics tells us that in equilibrium, any system will exist in the phase (or coexistence of phases) that minimizes the free energy. Hence, each point of the hyperplane of the relevant variables will be associated with a stable phase (or phase coexistence). These points and in particular the limits of stability of these phases can be collected, getting the *phase diagram*. Looking at many snapshots taken from different state points, we can observe three classes of possible configurations, which are collected in Fig. 2.3. In Fig. 2.3a we can clearly see the presence of areas with different densities, which we can associate with regions in a *liquid* and a (highly dilute) *gas* phase. In contrast, the other two other pictures present a homogeneous density in the whole system. While in Fig. 2.3b we observe that the bonds, which are actually a small number, does not form any particular pattern, in Fig. 2.3c many of the bonds are organized into preferential directions forming a hexagonal configuration, which is the most stable one with the kind of particles we are studying. We would conclude that the last snapshot, which bonds picture has been emphasized in Fig. 2.3d, represents in fact a phase separation, in which part of the system is fluid and part of the system is crystalline.

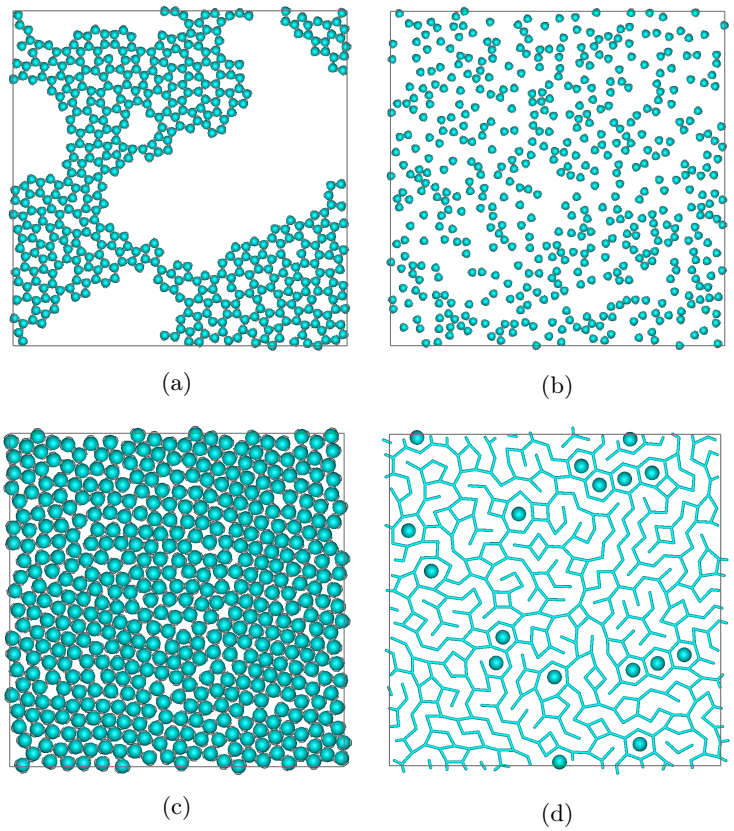
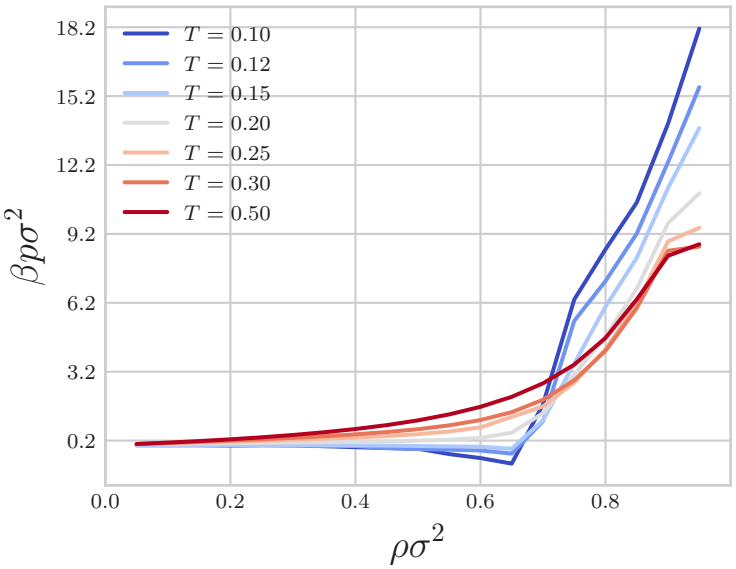


FIGURE 2.3: (a) Snapshot representing a phase separation, taken at $\rho\sigma^2 = 0.4$ and $\beta\epsilon = 10$. (b) Fluid phase, taken at $\rho\sigma^2 = 0.4$ and $\beta\epsilon = 10$. (c) Solid-fluid phase coexistence taken at $\rho\sigma^2 = 0.95$ and $\beta\epsilon = 6.36$. (d) Same snapshot of the system in the solid phase, but indicating the bonds between the particle in order to emphasize the preferential directions.

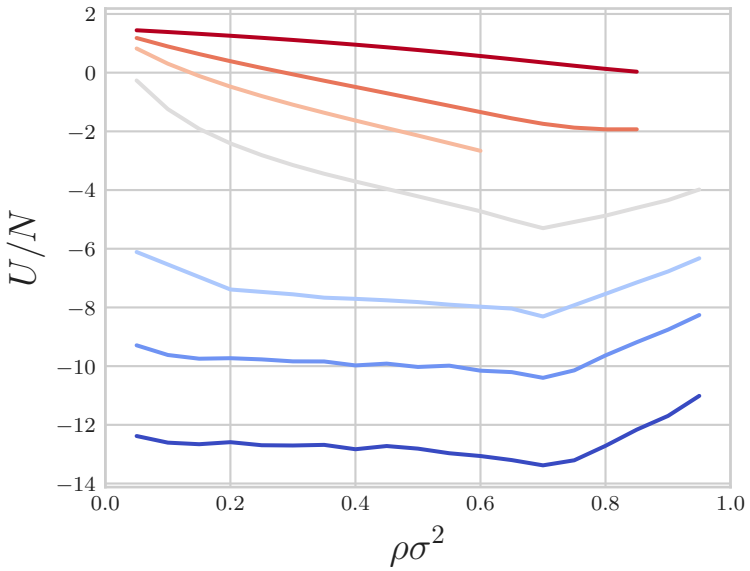
At this first stage, we can then separate the phase diagram into a fluid single phase region, a fluid phase separated region, and a crystal one. Obviously we could expect to be able to separate the gas and liquid phase, or even to find a liquid-solid phase separated region, but since their definition is not explicitly clear from the snapshots, we will partially neglect them at this point. Since the

state of the system can be inferred only by the global quantities describing its thermodynamics, we can plot the phase diagram in a $2-d$ plane as a function of $\rho\sigma^2$ and k_bT/ϵ. Each point would then correspond to the system at a specific density and temperature, and the other thermodynamic quantities will depend on them. The nature of these point can then graphically show in which region the system is stable in a certain phase (or coexistence of phases). By defining T_c as the maximum temperature at which we can encounter phase separation, everything in the single fluid phase region above T_c is simply a *fluid* (with no distinction between gas and liquid), while below T_c the single-phase fluid to the left of the phase separated region is the *gas*, and to the right of it is the *liquid*. In order to find the location of the regions, in principle we need to look at the free-energy minima calculated at each point. A system reaching the equilibrium will always tend to minimize its free energy, and hence the preferred phase will be the one corresponding to the minimum calculated at each point of the phase diagram. In this way it is easy to understand the coexistence of more phases, being it associated with the presence of two (or in principle more) competing global minima of the free energy. This analysis can be systematically done through the use of simulation techniques, with which it is possible to find the *phase transition curves*, which are the separation bounds between the different regions. Full free energy calculations are hard to perform in practice, because they are time consuming procedures. Instead a reasonable approximation of the phase diagram can be obtained simply by performing simulations at different state points and observing what phase forms. One straightforward way to do this is to follow an horizontal or vertical line in the phase diagram (*i. e.* keeping constant temperature or pressure) and look when a transition event occurs [43; 44]. Phase transformations can be detected by looking for evident variation of the behavior of the system. For instance, when the system transforms from one physical state to one other, the bonding pattern is expected to change, re-organizing the particles into a different structure. This phenomenon can then be detected e.g. via a variation in the potential energy and in the pressure of the system. This strategy has serious drawbacks when encountering hysteresis in the transition, usually present in first order ones, [45]. Hysteresis phenomena make it possible to find different phases at a given

state point depending on the history of the system (e.g. when gradually cooling it or gradually heating it). Hence, different initial conditions in a simulation may result in different phase. The reason for that is the fact that first order phase transitions present a free energy barrier separating the phases near the coexistence. The height of this barrier will depend on the formation of interfaces between the two separated phases. The larger the interface, and the larger the surface tension, the higher the barrier is, resulting in a more relevant hysteresis. This issue can be solved by studying the system without creating such interface, preparing it with an already existing interface from the beginning of the simulation or by eliminating its effect. There exist several methods performing this tasks, among which the most famous is the *Gibbs ensemble method* [46; 47]. We will not treat them in the detail since we only need information about the different phases at a lower degree of precision. Here, we simply perform an observation *by eye* of snapshots like the ones of Fig. 2.3, in order to assign them a label and have a simple separation of the regions of stability for the various phases. In order to collect more information about the position of the phase separation curves, it is then also possible to look at the equilibrium value of thermodynamic quantities the system reaches in the different points of the phase diagram. In Fig. 2.4 we plot for different temperatures the average values of pressure (Fig. 2.4a and potential energies (Fig. 2.4b) measured from the simulated system after equilibration.



(a)



(b)

Figure 2.4: (a) Dimensionless pressure measured in the system at variable density for different values of temperature. (b) Same analysis for the internal energy per particle.

From the analysis of the snapshots we are able to identify the presence of a phase separated region from very low density to density of the order of $\rho\sigma^2 \approx 0.6$, in a range which decreases as the system temperature increases. Over this region, the system behaves as a single fluid phase. Finally, at high pressure (around $\rho\sigma^2 \approx 0.95$) we observe the appearance of a solid phase. A qualitative phase diagram starting from these observations is shown in Fig. 2.5. In both the plots of figure 2.4 it is possible to observe a change in the first derivative for some temperatures at density around $\rho\sigma^2 \approx 0.6$, confirming the transition curve in this points. Moreover the curves in Fig. 2.4b also present analogous slope variations for low densities, giving a more precise instructions also for the position this curve. From Fig. 2.4a it is also possible to infer the position of another transition around densities of $\rho\sigma^2 \approx 0.7$, suggesting the fact that some clusters of the system in the solid phase could appear from this point.

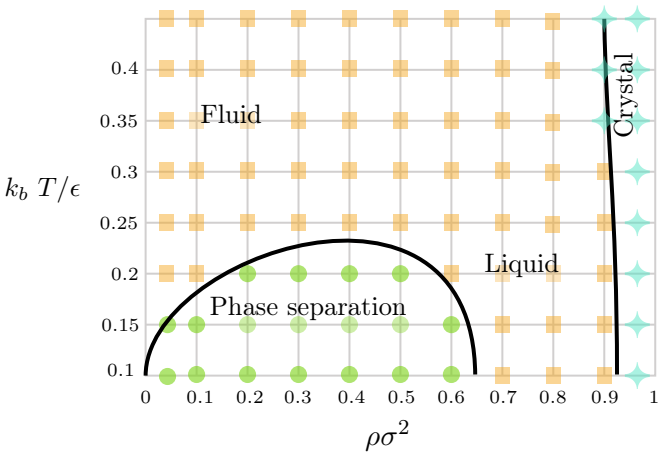


FIGURE 2.5: Phase diagram of the system. The regions has been found after a *by eye* classification of points of the plane in a grid.

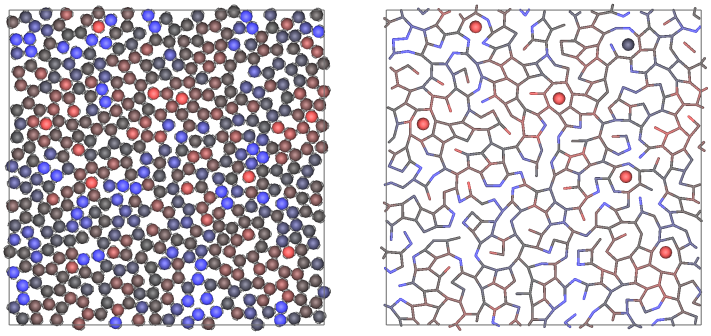
In this thesis, we are interested in studying the dynamics in a homogeneous fluid phase as a function of temperature. By looking at Fig. 2.5 we can observe that the ideal region for this task is the one with densities from 0.7 to 0.85, in which no phase separation or crystallization can be observed at any temperature. In the

following chapter we will then focus our attention on this region in order to analyze the local properties of the system.

3.1 Definition of the Problem

As explained in the previous chapter, the analytical study provided by Wertheim theory does not provide any information on the dynamical features of the system. In particular, due to the fact that all the bonds are considered in the same way, any attempt to use the same theory to predict bond dynamics would simply state that all of them are equally like to form or break. This approximation is obviously not sufficient to trace the evolution of the bonding patterns, because we expect that some regions, due to the local connectivity, may present different tendencies to change in terms of local structure. In order to check this assumption, we run a set of simulations from a fixed snapshot of our system equilibrated at dimensionless density $\rho\sigma^2 = 0.8$ and inverse temperature $\beta\epsilon = 5$. In each evolution we measured the time that each particle spent to run into a breaking/ forming event, and expressed the averaged result by accordingly coloring them as in Fig. 3.1. The particles with red colors are the slowest, while the blue are the fastest. All the times we will consider in this chapter are expressed in dimensionless units as $\hat{t} = t/\tau_0$, where τ_0 is the simulation time unit

$$\tau_0 = \sqrt{\beta m \sigma^2}. \tag{3.1}$$



(a) (b)

Figure 3.1: Snapshot of a system simulated at $\rho\sigma^2 = 0.8$ and $\beta\epsilon = 5$. Particles are colored to highlight differences in the averaged time of the first occurrence of a bond event for each particle, with red indicating longer times and blue shorter times. Snapshot (a) shows the particles, while in (b) the same snapshot is shown, but highlighting only bonds between particles (and a few unbonded particles).

Here we can observe that there are significant variations in the bond breaking and bond forming times in the system. Moreover, we observed that the gap between the fastest and the slowest particles increases when moving to lower temperatures and higher densities. This suggest that the presence of the rings, not considered in Wertheim theory, becomes more and more relevant in these conditions, and will play an important role in determining the rate for breaking and forming the bonds. One first observation to Fig. 3.1b shows that in our snapshot there are 6 particles which start with no bond, and 5 among them have a relatively long waiting time. This can be explained by observing that they are contained in closed ring, such that a bond formation can happen only after the breaking of one bond among the ones involved in the cycle. This is not the case for the 6th particle, which is contained in a cycle with another particle which has two unbonded patches. In this situation is reasonable to expect that a bond between the two can form more quickly.

This hand-waving reasoning can be easily applied to unbonded

particles, but it is a sort of hard to extend to the others starting from the information in Fig. 3.1. At this moment we are measuring in each simulation the lowest of the waiting times for each patches of a particles, without having a clear information on whether a bond forms or breaks. If we expect that the time of each bond forming/ breaking event is in some sense related to the presence of the rings, we need a way to evaluate their strength. This problem can be solved by extending this reasoning and measuring the waiting time for each patch on each particle separately. This also implies that we have to split the analysis into *breaking times*, involving the bonded patches, and the *forming times*, involving the unbonded ones. For its simplicity in visualization and treatment of the data, in the remainder of this thesis we will only consider the first class of events. However, we conjecture that an analogous reasoning can in principle be applied to the second. A color representation of the bond breaking time from a snapshot taken at the same conditions of the previous example is shown in Fig. 3.2.

In Fig. 3.3 it has been shown how the bond breaking time (t_b) can assume different values in different conditions. In this chapter we will develop a method to predict such variability starting from the topological differences of the system, both regarding the external conditions and the local structure. To perform this task we built a NN that will defined in the following sections.

3.2 Neural Network implementation

The main idea behind the construction of a NN is the possibility to create a program which is able to improve itself using natural brain-like learning processes. In a more general sense, a method which involves learning can be any program able to extract information from a set training samples in order to create classifiers, predictive functions or even new data from them [48]. For NNs, the key idea is to define an extremely general non-linear function, which is then trained to follow desired behavior. Training is basically done via an algorithm which aims to reduce the value of a certain cost function through many iterations on the training data. This reduction is performed by optimizing a set of parameters associated with the neural network, and in a broad sense of the underlying non-linear

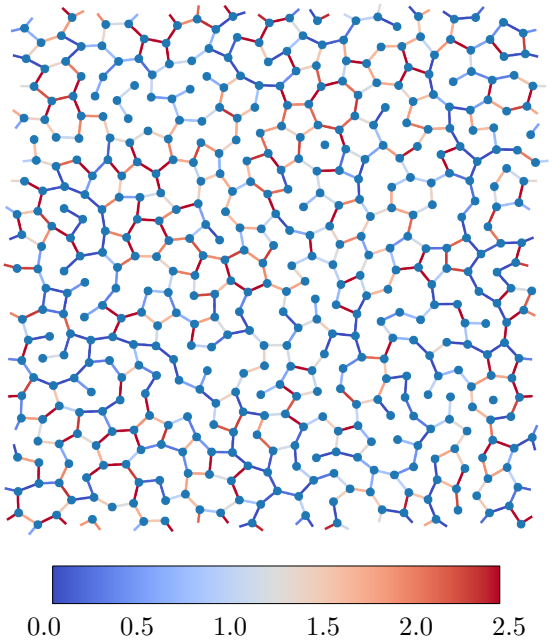


Figure 3.2: Snapshot of a system simulated at $\rho\sigma^2 = 0.75$ and $\beta\epsilon = 5$. Patches forming bonds between the particles are colored to highlight differences in their averaged bond breaking dimensionless time t_b/τ_0, with red indicating longer times and blue shorter times.

function. A NN is in fact nothing but a function, which can be written in the form:

$$\bar{y} = f(\mathbf{x}; \mathbf{\Omega}), \tag{3.2}$$

which takes the input values x and returns the answer $\bar{y}$. The parameters in Ω are the characteristic ones of the NN and they will be defined later. The way to reduce such cost is made by modifying the set of parameters of the non-linear function. A basic data set is a structure made of input points x from a d-dimensional hyperspace. The collection of these points will be recalled as X. Each point of this set is associated with a label. The meaning of this labels, collected into the set y, depends on

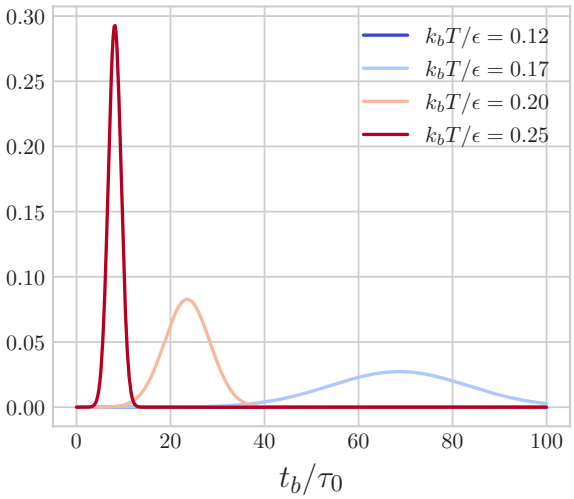


FIGURE 3.3: Gaussian fits to the distribution of the bond breaking times, in dimensionless units for different temperatures.

the type of learning algorithm but generally refers to the value that the network is supposed to return once trained.

First of all, the size of the space of possible outcomes assumed by the labels defines whether the problem is a *classification*, if the labels can assume only a discrete number of values, or a *regression*, if the algorithm has to infer a continuous value. In order to get into the framework it can be useful to look at how a Learning algorithm should work through one simple example. Let's consider a classification problem in which our problem has to predict one of two possible outcomes, for instance the words *"Yes"* and *"No"* from human speech. In this situation the set X will be a collections of recordings of peoples saying one of these two words, and for simplicity we can assume the possibility to extract just 2 relevant features from each of them, as for instance the amplitude of two harmonics in the Fourier transform of the signal. In other words, we want to condense all the information contained in the recorded files in two real-valued order parameters, which are obviously more treatable by a program. The data set

of this *toy model* of a *text-to-speech* problem will then be made of points in a $2d$ plane, each of them associated with a "correct answer" label that the algorithm has to predict. Recalling the *functional* representation from above, we will want to optimized $f(x_1, x_2, \Omega)$ such that it returns *"Yes"* on point with this label and *"No"* on the others. The data set is generally split into a *training set*, which the program will use to "learn", and a *test-set*, used to see if after the training it performs well on new data. One classifier which learns *too much* from the training set may expect to find some features that are contained in it without being representative of all the space of entering data. If this happens, the algorithm ran into an *overfitting* problem. We will say more about this in the following.

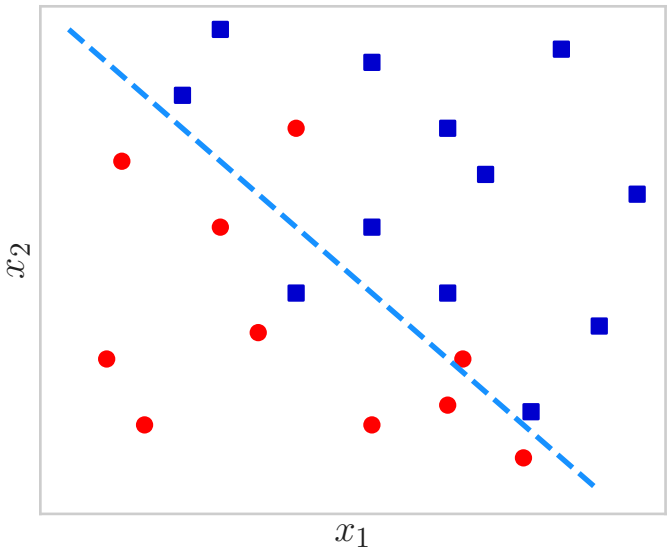


Figure 3.4: Red spots are the data with label *"Yes"*, blue squares with label *"No"*. The dashed line is the separation between the two classes inferred by the learning algorithm.

The general idea is use the training set to find a rule to divide the plane into K sub-planes, where K is the number of possible values the labels can assume, and then assigned a predicted label to the points depending on where they fall into the plane. In our example we can assume that the best possible way to separate the data set is to find a line. If a point falls to the left of it, the program will say that the answer is *"No"*, otherwise it will say *"Yes"*. The algorithm will then optimize the parameters of this line in order to minimize a cost function, which increases if the prediction is wrong and decrease if it is correct. For this example, a logical choice of cost function would simply be the fraction of points for which the label was incorrectly predicted.

After some iteration, in which the algorithm applies the current version of f on the *training set* and consequently update its parameters in order to make the cost function decrease, the program will find the optimal solution through linear regression like algorithms. While finding an optimal solution is likely possible in the simple linear example provided here, in practice the solution will just be locally optimal. Then in more complex cases the algorithm also needs to span more regions in the space of the solutions and it may be beneficial to make changes to the network that do not always decrease the cost function. In general the cost function is updated through the use of a GD like algorithm which is optimized in such a way to present a variable *learning rate* that permits both to move from one minima of the cost function to one other and to avoid large oscillation in the vicinity of a minimum.

From the example of Fig. 3.4 we can see that the solution the program found does not provide the correct answer to all the analyzed points, this is done to emphasize one of the main problem of the learning procedure: the risk of *overfitting*. A naive approach to the problem would be to substitute the divider in the figure with an higher order polynomial, *i.e.* increase the number of parameters to tune in our fitting function. This would allow us to completely separate the blue from the red spots. This correction, which in some cases can also lead to better results, is in general problematic because in this way we are complicating the model in order to make it exactly fit the training data, but we have no guarantee that it will remain good for any other new point. The reason for that, as can be easily visualized in the example, is that all the information in the spoken words has been condensed down

into only two real variables, meaning that a lot of information can be lost in the conversion. The resulting error can be seen as a noise that can even allow two points with the same values for (x_1, x_2) to have different labels. In other words we can say that each label can be primarily found in separated regions, but they follows gaussian-like distribution that can overlap in other regions. For a good choice of order parameters (x_1, x_2), the probability of being in such a region is low, but it is often not possible to fully eliminate it. One of the powers of *Machine Learning* (ML) techniques is actually their capacity to *find* the best solution even where analytical results may fail. Moreover, there are several common ways to avoid overfitting, such as keeping the number of parameters in the model relatively low and limit the number of iterations. Such *hand-waving* rules can actually be made more strict when looking at a specific case. In the following we will outline the main types of ML algorithms and the broad types of problem they typically aim to solve.

Unsupervised Learning — In *unsupervised learning* the labels of the input data are not predetermined and so the algorithm has to understand the best grouping by itself (*i.e.* there is no *teacher*). What can often be defined a priori is the number of groups and some of the rules the program should use for the clusters. An example problem solvable with this technique is the classifications of news in an online provider, such as *Google News*. Here there is no a priori label for the incoming news, since no one knows the day before what will happen. The goal of the program is to find the best clustering of the different information, in order or collect the ones talking about the same topic.

Supervised Learning — In *supervised learning* we do have a teacher, so each input comes with a *true* desired output and the algorithm has to find the underlying rule to assign labels. The example previously discussed is a typical problem of supervised classification. As previously said, the space of possible answers can also be continuous. In the case, then the goal is to find a function, typically getting a real number from all the order parameters. This type of regression problem is exactly the one we want to solve in this thesis.

Reinforcement Learning — A slightly different approach, applicable to classification problems, is the *reinforcement learning*. Here we still have a data set composed of points with label, but the *teacher* plays the role of just saying if a predicted solution is right or not. This means that the only feedback the algorithm receives is a *critic*, a binary signal with no information on the correct classification if the given answer is wrong (obviously if the possible classifications are just two, supervised and reinforcement learning are equivalent). The idea behind this way of learning is the fact that the program have to understand not only how to decide, but also which parameters are more important in this task.

3.2.1 What is a Neural Network?

Focusing on supervised learning, there is a vast collection of possible methods for designing and training the classifiers. One of the most widespread and in some sense most *natural* ways of learning are provided by the use of so called artificial NNs [48]. A NN (fig. 3.5) is a series of connected units, the *neurons*, usually separated into layers. Each neuron is able to produce an output signal based on its inputs arriving from the previous layer using some characteristic weights, which are the parameters that have to be optimized in the learning process. In principle, a multilayered network of this kind can in principle solve - which means find the relation minimizing a certain cost function - any classification or regression problem thanks to the *Universal approximation theorem* [49]. A clever choice of such cost function and of parameters properly describing the system under study are then fundamental to make this solution close to the real solution of the problem. The key power of a NN is not a magical ability to understand the data given to it, but simply lies in its possibility to use simple algorithms to catch and extract the non-linearity of the solution from the training data. One of the most widespread methods to accomplish this is *backpropagation*, which calculates the error derivatives with respect to the weights, in order to apply then GD like procedures. This algorithm has a relatively simple theory, but it is generally implemented with a series of tricks used to increase performance and training speed. Another point of primary importance in the construction of the network is the attention to dealing with the

model selection (meaning the complexity of the network) and the regularization. The parameters entering in the NN network implicitly reflect the nature of the system, because they are the ones chosen to feed the network, but this is no longer true when looking at the hidden layers. In fact it is possible to chose an arbitrary number of them made of any number of neurons, with consequent no constraint at the complexity of the model. This is problematic because if the network is too simple, it would not be able to catch all the features of the system, but if it is too complex, it could easily suffer of *overfitting* problems. Deciding the right level of complexity is then crucial in getting the best result, and in some way this can be inspired by the model we want to predict. Even though research on this field has been ongoing for decades [50; 51], there is no perfect recipe for deciding how to design the optimal network for a given problem. Although some guidelines exist, the network features will need to be adapted to the question under consideration.

The algorithmic idea behind a neural network is to iterate a process that modifies the weights on the neurons and verifies the effect on the cost function. While the *backpropagation* is used to find the best way to update the weights *backwards* as a feedback of the previous ones, each step make also a *forward* passage through the network, which is the *feedforward operation*. Both these element of the algorithm are schematically shown in Fig. 3.5 as arrows following the respective directions.

3.2.1.1 Feedforward passage

The example NN of Fig. 3.5 shows n neurons, labelled as x, in the input layer, representing the vector of n order parameters used to characterize the system. Each of them is is connected with all the neurons of the first hidden layer (m_1 neurons labelled as a), which are interconnected with all the neurons of the second hidden layer. The network can have an arbitrary number of hidden layers, and the last produces the output (here only a neuron, representing the quantity that has to be predicted). Each connection in Fig. 3.5 is associated with a modifiable weight, indicating how strongly the output of the source neuron affects the output of the target one. More specifically each neuron represents a function that linearly combines the weighted inputs from the previous layer, getting for

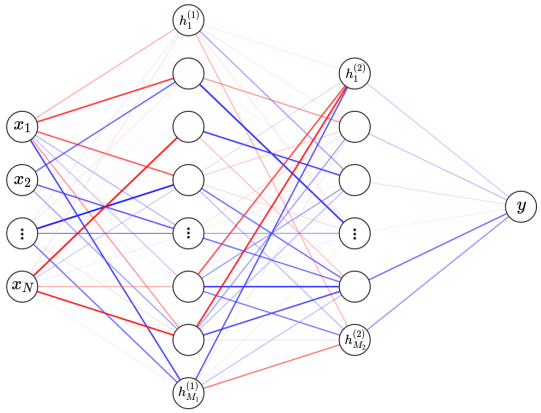


FIGURE 3.5: Basic structure of a NN. The network presented here has an input layer, an output layer, and two hidden layers between them. The connections between the layers are represented with different colors and thicknesses in orther to emphasize the fact that they have different weights.

the first hidden layer:

$$a'_j = \sum_{i=1}^{n} x_i w_{i,j}^{(1)} + w_{j,0}^{(1)} \equiv (\mathbf{w}_j^{(1)})^t \mathbf{x}. \tag{3.3}$$

In the previous equation can be observed the presence of the parameter $w_{j,0}^{(1)}$, which is the bias of the linear function of the neuron. This extra parameter can be thought as a weight by adding one more fictitious input parameter x_0 with value 1. In this way it is possible to define the vectors $\mathbf{w}_j^{(1)}$ and $\mathbf{x}$, respectively as the vector collecting all the weights and the one containing the order parameters and x_0 as entries. Then the summation can be written as a scalar product, as in the last equivalence of Eq. 3.3. To make again a comparison with a physiological brain, these interconnection between the neurons carrying the stimuli are nothing but *synapses*. Hence, the values w can be called *synaptic weights*. Instead of emitting exactly the signal a'_j, each output of the hidden unit is generally a nonlinear function of it, such that the signal provided by a neuron to the outgoing synapses is given

by:

$$a_j = \phi(a'_j). \tag{3.4}$$

In this equation $f(\cdot)$ is the activation function of the unit, which can in principle be different for each layer or even for each neuron. A NN constructed without this correction would be a linear function of the entries. Then any added layer would simply make more complex the relations for the parameters of the output, with no possibility of changing its structure. Obviously such network would be ineffective for fitting most of the possible functions it may have to predict.

One more reason for adding this function can again be taken from the physiological features of the neurons. In a first approximation we can think of each unit as having just two possible states, excited or not, depending on the intensity of the entering signal. In other contexts it may be useful that an output is always positive, or should be necessary to prevent it to increase to much as the input increase. To summarize up, depending on the problem we want to analyze, or even for stability reasons, it is generally useful to pre-process each signal in order to provide it features that can not come from a linear combination. Moreover, the backpropagation step will require a derivative of the signals related to the inputs, because it is necessary to keep a sort of track of where the most important signals come from. In Tab. 3.1 there are briefly presented some of the most common transfer functions (also called *activation functions*).

The composition of this function and the linear combination of the entries is known under the name of *Linear Perceptron* (LP). The NN we are now analyzing consist of more layers connected together, then it is called *Multi-layer Perceptron* (MLP). One of the major advantages of using more layers instead of just one is that it makes the network much more effective at fitting a nonlinear behavior. The use of *deep* (*i.e.* made with more layers) NNs has been demonstrated to be much more efficient to train in fitting complex data, being able to distinguish even not linearly separable data [52]. These advantages gave raise in the last two decades to the popularity of the so called *Deep learning*.

The approach of Eq.3.4 can then be applied to all layers,

Function	Expression	Features
Heaviside step function	$\phi(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	Binary signal
Sigmoid function	$\phi(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	Preventing "jumps" in output values.
Hyperbolic tangent	$\phi(x) = \tanh(x)$	Same features as Sigmoid but symmetric and centered in zero.
ReLU (Rectified linear unit)	$\phi(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	Simple but still having a derivative dependent on the input.
ELU (Exponential linear unit)	$\phi(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	Same as ReLU but significant also for values approaching zero

Table 3.1: Most common activation functions.

getting:

$$\begin{aligned} b_k &= \phi\left((\mathbf{w}_k^{(2)})^t \mathbf{a}\right) = \phi(b'_k) \\ y &= \phi\left((\mathbf{w}^{(3)})^t \mathbf{b}\right) = \phi(y'). \end{aligned} \tag{3.5}$$

3.2.1.2 Backpropagation algorithm

Backpropagation is the part of the algorithm which permit to give a feedback to the learned parameter every time they are improved. It can be seen as a generalization of a *Least mean square* (LMS)

analysis that works for linear systems. In the *supervised learning* approach we are looking at, we have a cost function which have to be related to the error committed comparing the predicted value and the real ones. In the following we will start by briefly explaining the logic of this algorithm applied to a network with just two layers and a number M of outputs. The same reasoning can be extended to a larger number of layers. Here we would have for $k \in \{1 : M\}$ as in Eq. 3.4

$$y_k = \phi\left((\mathbf{w}_k)^t \mathbf{x}\right). \tag{3.6}$$

One natural choice of the cost function will then be the LMS one:

$$J(\mathbf{w}) := \frac{1}{2}\sum_{k=1}^{N}(\bar{y}_k - y_k(\mathbf{w}_k))^2 = \frac{1}{2}\left(\bar{\mathbf{y}} - \mathbf{y}(\mathbf{w})\right)^2, \tag{3.7}$$

where the dependence of J by the weights of the network in emphasize by introducing $\mathbf{w}$ as the vector containing the weights of the layer. In order to minimize J, the weights have to be modified following the direction reducing the error. In other words they have to be changed of an amount proportional to the inverse of its partial derivative with respect to the weight:

$$\Delta\mathbf{w} = -\alpha\frac{\partial J}{\partial \mathbf{w}}, \tag{3.8}$$

which in components means

$$\Delta w_{i,k} = -\alpha\frac{\partial J}{\partial w_{i,k}}. \tag{3.9}$$

The parameter α is the *learning rate*, and is a measure of how much each weight can change. Obviously a clever choice of it, which can also vary during the training procedure. is fundamental to get ion good results. Each step of the computation then will simply update the weights from the previous as

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \Delta\mathbf{w}^t. \tag{3.10}$$

Adding more layers, as in the example of Fig. 3.5 , the extension straightforwardly obtained by applying the chain rule to the

derivatives. Going deeper into the evaluation of the derivatives, the last series of weight derivatives will be:

$$-\frac{\partial J}{\partial w_k^{(3)}} = -\frac{\partial J}{\partial y'}\frac{\partial y'}{\partial w_k^{(3)}} := \delta_k^{(3)}\frac{\partial y'}{\partial w_k^{(3)}} = (y-\bar{y})\phi'(y)b_k, \quad (3.11)$$

where the two derivative are evaluated respectively from Eq. 3.5 and Eq. 3.7. We have also defined the *sensitivity* of the layer as $\delta_k^{(3)} = -\frac{\partial J}{\partial y'}$. Being the derivative with respect to the signal entering the activation function, this parameter in fact measure how the error changes with the neuron activation. With the same procedure the other partial derivatives can be evaluated going backward as follows:

$$\frac{\partial J}{\partial w_{i,k}^{(2)}} = \frac{\partial J}{\partial y'}\frac{\partial y'}{\partial b_k}\frac{\partial b_k}{\partial w_{i,k}^{(2)}} = (y-\bar{y})\phi'(y)\phi'(b_k)a_i \quad (3.12)$$

3.2.1.3 Learning step

Once we have defined the procedures forming each computational step, we need to understand how to effectively feed the network with the *training set* and perform the improvements of the weights. There are substantially tree GD like protocols to perform this task, which mainly differ from each other for the portion of data presented to the network at each step[53]. To summarize up the passages done at this point we can see the learning procedure as in Alg. 1. Here the NN with all the weights is represented through the function $f(\cdot)$. The GD variant that we are going to introduce will then define the strategy to define the subset $\{x\}$ of the training set in line 2 and the way of updating the weights in line 4.

Algorithm 1 Learning step

for $t \in [0:T]$ **do**
  $y \leftarrow f(\{x\})$ ▷ Calculates the output through feedforward
  $\nabla J \leftarrow$ backpropagation$(J, \mathbf{w})$
  $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$

Batch GD — This procedure computes the gradients of the cost function for all the element of the training set together at

each step, then in Alg. 1 we will have $\{x\}$ as the whole set. The fact that for *each* update it is necessary to perform the derivative for all the input data makes this strategy potentially very slow and needs a large amount of memory. This can be done by defining the total error J as the sum of the errors committed by the network for all the *training examples* contained in the input set.

Stochastic GD — The GD is performed for one *training example* at each step. This means that at each step just one *pattern* is taken at random from the training set, as if it would be a random variable drawn from the unknown distribution the network has to learn (from this the name *stochastic*). One one hand, this method avoids the redundancies in the computation of the gradients we would have in a Batch GD, making each step much faster. Moreover the different example can be taken *online*, meaning that they can even be generated one at time during the training itself. On the other hand, each pattern, and then the updates, can present a high variance which cause the presence of heavy fluctuations in the values of the cost function that gets updated.

Mini-batch GD — A strategy in the middle of the previous two is Mini-batch GD, acting on both reducing the variance of the updates and the dimension of the pattern shown at each step. This can be done by separating the whole training set into smaller subset, then *presenting* one of them at a time to the NN.

Since these methods does not guarantee a good convergence in all cases, people usually associate them with the use of *optimization algorithms*. There are in fact many points can be looked with more attention in the definition of the weight update. One among the other is the introduction of a variable learning rate α [54]. We can think this parameter as a measure of the distance the network can travel in the space of the solutions during one single update. For sure we can expect that an high rate at the beginning of the computation can be a good choice to make the system able to space more region in search of the minima of the cost function. On the other hand, in the proximity of a minimum one

weight too high would make the NN oscillate around it, without reaching a convergence. The direction followed by the system in the minimization procedure without a proper choice of the learning rate can lead to some strange behaviors, like the one schematized in Fig. 3.6. When the system encounters these saddle-like configuration of the cost function, a *naive* application of the GD algorithm would make the *ball* follow a path like the red one. In fact every time the gradients are performed on the top of the hill, the preferential direction is the one to the center of the saddle, but a too long movement will drive the ball to the opposite side of the valley. Iterating this step the network would eventually reach the minima, but following a path that is obviously not the optimal one. Also in this kind of situations the use of an *optimizer* would make the system *smarter*, becoming in some sense able to understand the configuration in which it is located and modify α accordingly.

There is actually a broad range of possible optimization function which can solve these challenges. The one we will use in our algorithm is known under the name of *Adaptive Moment Estimation* (ADAM) [55]. The main idea of the algorithm is to keep trace of the path the ball is following when it is going down, and modify it before it goes up again. This is practically done by storing the first order gradient in order to estimate their first and second moments, used to compute different learning rates for each parameters.

3.2.2 Characterization of the system

As previously described a Neural Network can be used to predict a property of the system starting from some input parameters. The goal is then to find some clever set of order parameters which could be able to characterize each bond. As observed in 3.2 the expected breaking time of a bond is strongly related to the presence of rings involving it. For that reason a useful starting point could be to collect the lengths of the rings for each bond. Remembering that each particle can have at most 3 nearest neighbors, each edge can be associated with a maximum of 4 loops: 2 involving the edge itself and one extra loop for each of its vertices, as can be seen in fig. 3.7. Moreover, while the rings are a suitable starting point, any other parameter characterizing the local structure of

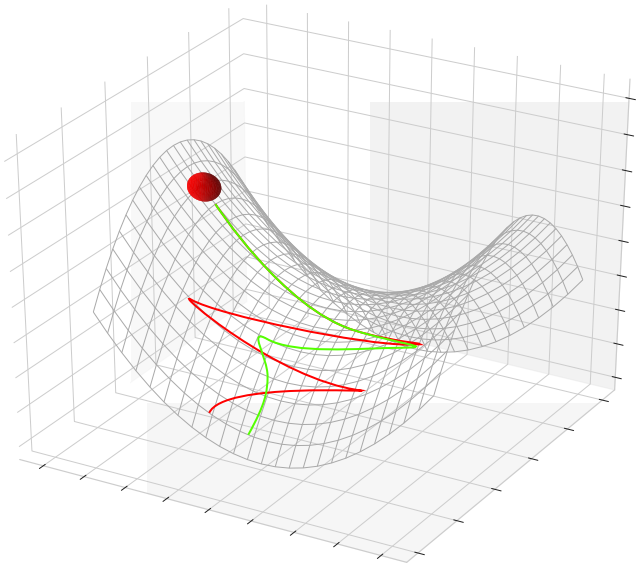


Figure 3.6: Graphical representation of the NN, schematized as a ball on the potential, encountering a saddle point in the space of the solutions. The red path represents a naive application of a GD-*like* algorithm. The green path is instead the result of an optimizer on the algorithm.

the network can in principle be a good candidate for the task. In the following, we will present all order parameters we employed to characterize the local structure around an edge.

3.2.2.1 Order Parameters

Our neural network aims to predict an expected breaking time for each bond. As its input, it accepts a set of order parameters describing the local structure around a single bond. The particle system can be considered as an undirected graph, meaning that each bond (i, j) which is present in the graph comes with its reverse (j, i). This implies that the edge-list can be simply contain

sets of two points in which the order is not important. Despite that, in order to give an univocal convention, each edge (i, j) is taken as if i is the smaller and j is the bigger of the two involved vertices. Then for each edge k, we define a vector $\mathbf{x}_k$ containing the set of the related order parameters. This set consists of:

Involved cycles — In principle each edge can be involved into at most 2 rings, one to the left and one to the right of it. They are found by iterating an algorithm finding the next connected vertex in the graph, consistently turning left or right with the use of geometrical information of the bonds. The algorithm stops when it comes back to the starting vertex, collecting all the encountered vertices. After this process it is possible that the loop passes over one vertex multiple times, meaning that the program enters one *impasse* and comes back. In order to exclude these passages, which actually are not part of the searched cycle, the loop has to be cleaned, deleting all the touched vertices between two occurrences of the same one, obviously excluding the starting point. All the rings with length smaller than 3 at the end of this process have to be neglected, since the are just the edge itself.

Moreover, in order to avoid extremely large ring sizes to carry large weights in the neural network, we set the length of any ring containing more than 16 particles to 16. Non-existing rings are set to the same value, as a not closed ring could be seen as an infinitely long one.

Extra rings — When considering the edge (i, j), if the particle i is also connected with two other vertices k and l, there may exist up to one ring more with respect to the ones previously considered, passing through l, i, k. The same is true for particle j, allowing for a total of up to two additional rings. These extra ones have been obtained using the same algorithm as before. The rings detection has been accelerated by the use of a *dictionary*, a data structure which saves all previously found rings and hence avoids considering the same ring multiple times. The basic version of the function used to find the rings is reported in Alg. 2. The function requires a direction (left or right) in order to find the right

one cycle. In order to store all rings correctly, the convention used is to name the left ring involving $i - j$ (with $i < j$) as the *first* (r^1_{ij}) one and the *right* as the *second* one (r^2_{ij}). In this way, once a ring is collected, it can be stored in the correct position for each of the edges present by keeping the correct direction for each. In this way the extra cycle connected with i would be the one to the *left* of the edge connecting i with its neighbor k in $\{k : r^1_{ij} \cap \partial i, k \neq j\}$ if $i > k$, or the *right* one in the reverse. The same reasoning can be done for the eventual extra ring of j. In Alg. 2 the function `bondafter(·)` in the code looks for the edge in the chosen direction with respect to $i-j$ by looking at the actual position of the particles, while `cleancycle(·)` excludes from the loop all the overcounted edges as described before.

Algorithm 2 Cycle counter

```
function CYCLE(i, j, direction)
    if {{i, j}, direction} ∈ Dict then
        return Dict({{i, j}, direction})

    loop ← [i, j]
    while loop[−1] ≠ loop[0] do   ▷ Stops when closes the loop
        loop.append(bondafter(loop[−1], loop[−2], direction))
    loop ← cleancycle(loop)

    if len(loop) < 3 then      ▷ Loops of smaller length are not rings
        return Dict({{i, j}, direction}) = {·}

    for e ∈ loop do   ▷ Spans over all edges {e_i, e_j} in the loop
        reverse direction if e_i > e_j

        if {e, direction} ∉ Dict then
            Dict({e, direction}) = {loop}
    return Dict({{i, j}, direction})
```

Bonded Neighbors — two numbers from 1 to 3 indicating how

many patches on both i and j are involved in a bond.

Average second nearest neighbors — Two parameters collecting for both i and j the average number of bonds their neighbors are involved in. This results in real numbers between 1 and 3 measuring the local connectivity of the system.

Nearby particles — Tree parameters indicating for differed radii how many particles are located in a circular region centered in the mid-point of i, j, regardless of whether these particles are bonded or not.

Convexity — Two binary values stating if the first two cycles involving the edge are fully convex. These are calculated by looking at the angle between the new added edges and the previous one in a ring when it has been found.

Angle deformation — Four numbers indicating how much the angles between the edge i, j and the one formed with the neighbors differs from $2/3\,\pi$. These last order parameters come from the idea that the preferred angle between two consecutive bonds in a ring is $2/3\pi$. Hence, the expectation is that large deviations from this angle will indicate rings that are under stress, and hence bonds that are likely to be broken early. We can resume this order parameter for the kth edge with angle $\theta^{(k)}$ as:

$$x^{(k)}_{\text{deform}} = \theta^{(k)} - \frac{2\pi}{3} \tag{3.13}$$

One first analysis that can be done on these data is the study of the correlation between them. While not being a necessary condition, an high linear correlation between an order parameter and the target time would imply that such parameter is more relevant in the prediction. On the other hand, two order parameters highly correlated between each others would in principle share the same information. This issue actually has minor effects on the predictions when using a NN, since the use of more layers allow the algorithm to catch features even from input data of this kind.

In Fig. 3.8, we collect a graphic representation through an *heatmap* of the correlation between the order parameters and

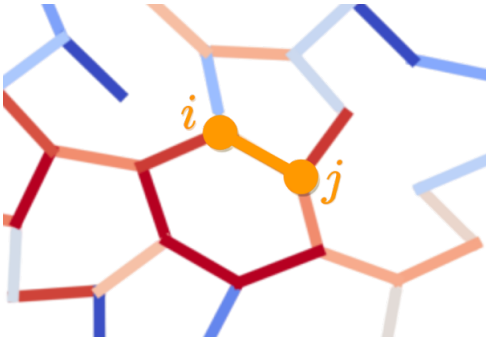


Figure 3.7: Cycles associated with the edge (i, j)

the target and between each others. The diagonal of the matrix obviously contains the maximum value, since they represent correlation between the same object. As we can see, none of the parameters has a correlation with the target time (the last line) with modulus higher that $0.2 \sim 0.3$. This order of magnitude for the correlation is relatively low, meaning that a significant prediction can be obtained only by looking at all the information we have, because all the order parameters taken separately are not sufficient to catch the target behavior.

3.2.2.2 Breaking time calculation

All the points in the data set are collected starting from one *snapshot* of the system taken at a specific time, after equilibration. In order to obtain a good statistics of the breaking times, the common strategy would be to take many results and averaging among them, *i. e.* simulating the evolution of the system many times from the picture. This idea is obviously a pretty long and computationally heavy task, since each repetition of the process has to last sufficient time to let all the bonds break at least one time. Moreover, this approach has the clear downside to relate the breaking of the first bond with all the ones after, because it influences the structure of the system. Since we are interested in the bond breaking time given a specific network structure, this undesired effect is definitively a problem to avoid.

A more clever idea is to take just one simulation, but modify

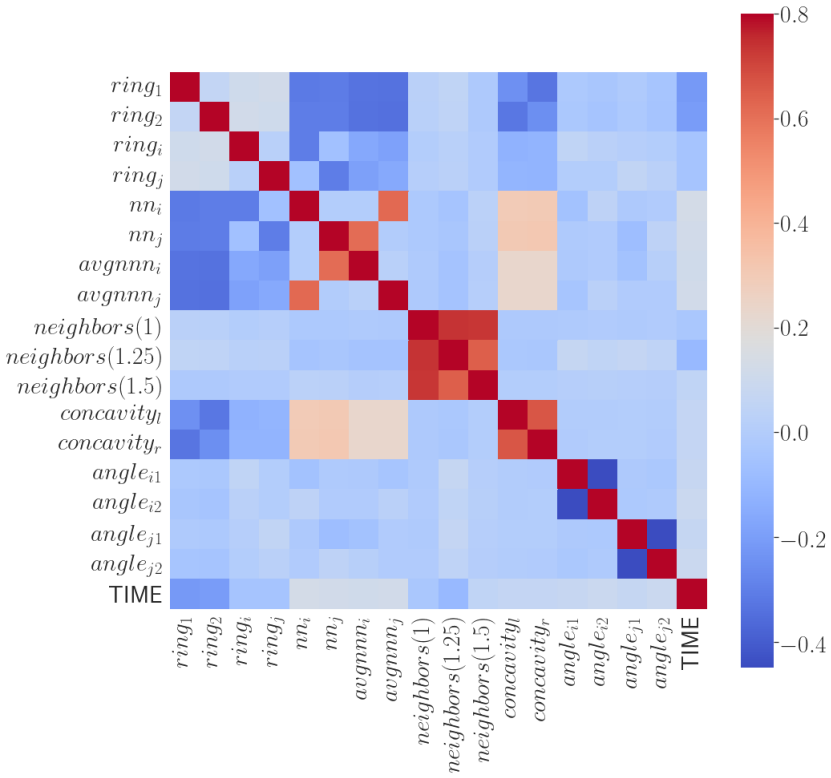


FIGURE 3.8: Heatmap of the linear correlation between the order parameters of the neural network and the target. The data are for $\rho\sigma^2 = 0.75$ and $\beta\epsilon = 0.5$.

the system in such a way that it is able to store each bond-breaking event without actually changing the bonding configuration in the system. In other words, we can perform an EDMD simulation in which the breaking of a bond (or the formation of a new one) is prevented by an infinitely high potential energy barrier, and we simply count how often a bond would have broken if that potential energy barrier were not there. Then, under the assumption that energy associated with the bonds is much higher than the kinetic energy, which is the more valid the lower is the temperature, it is possible to assume that the system configuration evolving in

such way always remains the same, even if the particles move from their starting position. In other word we can say that the system status is totally contained in the bonding pattern, while the instantaneous particle positions have only a minor effect.

With this approach each simulated time between one bond breaking event and the next one can be considered as an independent measurement of a bond breaking time, and we can obtain statistics on the expected breaking time for each bond from a single simulation. Using a fixed simulation time t_{sim}, the expected breaking time t_b for each bond present in the starting pattern can be computed through the number of occurrences n_b of a breaking event as follows:

$$t_b = \left(\frac{n_b}{t_{\text{sim}}} \right)^{-1} \tag{3.14}$$

At sufficiently low temperatures, we expect the breaking of bonds to be a *rare event*. In this regime, we can assume that the breaking time for each bond follows an exponential distribution, with a different decay time for each edge, corresponding to the mean of the distribution. If t_{sim} is sufficiently large, it will be considered exactly as the total time of n_b samples from this distribution, then t_b calculated in this way is just the average as

$$t_b = \frac{t_{\text{sim}}}{n_b} \approx \frac{t_1 + t_2 + ... + t_{n_b}}{n_b} = \langle t \rangle. \tag{3.15}$$

Then for each bond k, t will be sampled from

$$P^k(t) = \frac{1}{t_b^k} e^{-(t/t_b^k)}. \tag{3.16}$$

The accuracy of our estimate for t_b can be evaluated by considering the expected error we would obtain if we measured t_b by drawing n_b samples from the distribution of Eq. 3.16 and taking the mean. Via the Law of Large Numbers, this error will scale with the inverse square root of the number of samples. Recalling that the standard deviation of an exponential distribution is equal to its average, we obtain a standard error:

$$\sigma_{\text{err}} = \frac{t_b}{\sqrt{n_b}} = \frac{t_{\text{sim}}}{n_b^{3/2}} \tag{3.17}$$

3.2.3 Implementation of the network

The neural network has been implemented with a python program, through the use of *PyTorch* libraries [56]. This deep learning framework is a powerful instrument that allows for efficient calculations by the use of NumPy-like operations and structures, and can be accelerated using the strong parallel computation power of GPU cards. PyTorch has found widespread use due to its simplicity, efficiency, and open-source nature. The commands defining a NN straightforwardly reflect the theoretical structure described in the previous chapters, and the implementation is generally intuitive [57]. The building blocks necessary to construct the NN in PyTorch are the following:

Data preparation — the data set has to be converted in a *pytorch tensor* to be used. This data structure can also be treated using GPU, then easily allows to switch between GPU and CPU, in order to improve the performances.

Loss function — Since the algorithm starts with random weight attributes, the predicted result can be at any arbitrary distance from the real value. then the measure of such prediction we used is a *Mean Square Error* loss. This is simply obtained at each steps by summing up all the squares of the differences between the prediction and the targets.

Optimizer — The optimizer is used to better perform the update of the weight. We chose the previously introduced *Adam* optimizer.

Network definition — The network can be defined as the series of the layers composing it. Each layer has been defined as a LP, with a linear combination of the entries and a subsequent activation function. An example 4 layer NN definition is introduced in 3. The layers have respectively N, $M1$, $M2$ and 1 neurons each, while the activation function is the ELU previously introduced.

Algorithm 3 Neural Network

```
net = torch.nn.Sequential(
    torch.nn.Linear(N, M1),
    torch.nn.ELU(),
    torch.nn.Linear(M1, M2),
    torch.nn.ELU(),
    torch.nn.Linear(M2, 1)
)
```

Loading Data — In order to deal with the input set, Pytorch provides an utility that allows to automatically shuffle the elements, batching the data, and eventually use *multiprocessing workers* to load them in parallel. In this way we are able to perform a Mini-batch GD on each learning step.

In Alg. 4 we report a simplified version of the algorithm we used to train the network in order to emphasize these elements.

Algorithm 4 Learning Procedure

```

Loader = Data.DataLoader(
    dataset=dataset,
     batch_size=BATCH_SIZE, shuffle=True )

#Optimizer: Adam
optimizer = torch.optim.Adam(net.parameters(),
    lr=0.0025,
    weight_decay=0.00001)

for t in range(T):
    for (batch_x, batch_y) in Loader:

        prediction = net(batch_x)     # input x and
    predict based on x

        loss = loss_func(prediction, batch_y)

        optimizer.zero_grad()   # clear gradients for
    next train
        loss.backward()         # backpropagation and
    gradients
        optimizer.step()        # apply gradients
    pred_test = net(X_test)
```

3.3 Results

We apply the NN to systems simulated at different values of temperature and density, observing different conditions of the system in the single liquid phase. We preferred to not analyze for the moment crystallization and phase separation events, since a longer time for equilibration and a slightly bigger data set would be probably required to get sensible predictions. Hence, we focus only on state points where the system remains in a homogeneous fluid state. Nonetheless a learning approach can in principle be applied to all the state points of our system.

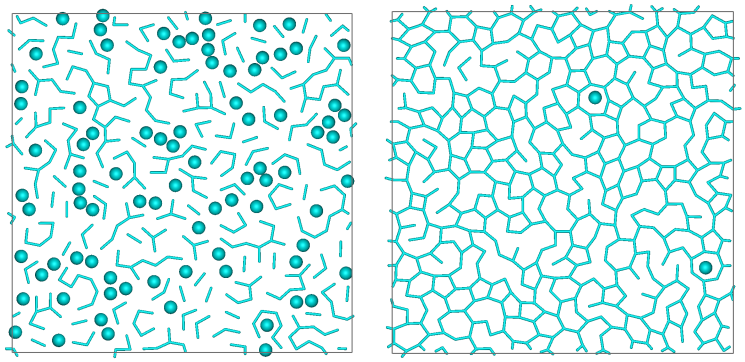
We generated data for density of $\rho\sigma^2 = 0.75$ and 0.8, in order to have a significant presence of rings while still avoiding the formation of solid state clusters. In order to measure the accuracy of each test, we make use of the linear correlation coefficient, which

expresses how much the predicted values are linearly correlated with the real ones. The so called $\mathcal{R} - correlation$ function is defined as:

$$acc = \mathcal{R} = \frac{\sum(y_p - \langle y_p \rangle)(y_r - \langle y_r \rangle)}{\sqrt{\sum(y_p - \langle y_p \rangle)^2 \sum(y_r - \langle y_r \rangle)^2}}. \tag{3.18}$$

For simplicity of notation, the subscript in the summation has been omitted: the sums span actually all the elements of the vectors y_p and y_r. This quantity is normalized in such a way that it can take values in the range $[-1, +1]$, where the extremes mean total correlation or total anti-correlation. Since we do not expect the net to produce predictions anti-correlated to the real values, we assume this quantity to be positive at the end of the training, and the higher it is, the more accurate is the prediction.

Varying the temperature we see different behaviors of the NN, both in terms of accuracy and in terms of resulting predicted values. In the following we analyze the prediction for two state points, at high and low temperature, in order to emphasize these peculiarities.



(a) (b)

Figure 3.9: Snapshots of the system taken at density $\rho\sigma^2 = 0.75$ and respectively $\beta\epsilon = 200$ (a) and $\beta\epsilon = 500$ (b).

3.3.1 High temperature

When studying the system at temperature relatively high (here $\frac{k_b T}{\epsilon} = 0.5$), we observe the presence of many small clusters of particles and rather few rings. The obtained distributions for the order parameters are summarized up in Fig. 3.10.

Here we can see that, as observed in Fig. 3.9a, most of the particles are not involved in any ring, as the distribution of the first four order parameters are highly peaked around the default value of 16. One other interesting remark is the fact that most of the rings are convex, which is reasonable because the small rings are more likely to assume regular shapes.

In Fig. 3.11 are plotted the data predicted by the trained NN over the real values. The red dotted line represents the ideal result (namely in the $x - y$ plane the line $y = x$), such that the more the result are adherent with it, the better the prediction is. For completeness the graph also shows the upper and lower bounds in accordance with the statistical error of Eq. 3.17. Looking at the times obtained in this way, a prediction is reasonably true if contained within a range of $2\sigma_{\text{err}}$ from the exact line, where (from rearranging Eq. 3.17) $\sigma_{\text{err}}(t_b)$ is:

$$\sigma_{\text{err}}(t_b) = \frac{t_b^{3/2}}{\sqrt{t_{\text{sim}}}} = \frac{(\langle t_b \rangle \cdot y)^{3/2}}{\langle t_b \rangle \sqrt{t_{\text{sim}}}} \tag{3.19}$$

where in the second equality σ_{err} is expressed as a function of the normalized time y. In Fig. 3.11, the orange spot represent the training set, while the blue ones are the test set. The fact that the two populations have almost the same distribution is a reasonable starting point to say that the network does not run into over-fitting problems. All the times from now on are measured in a normalized scale, defined as $t_{\text{real}} := t_b/\tau_b$, where τ_b is the averaged time for each measure.

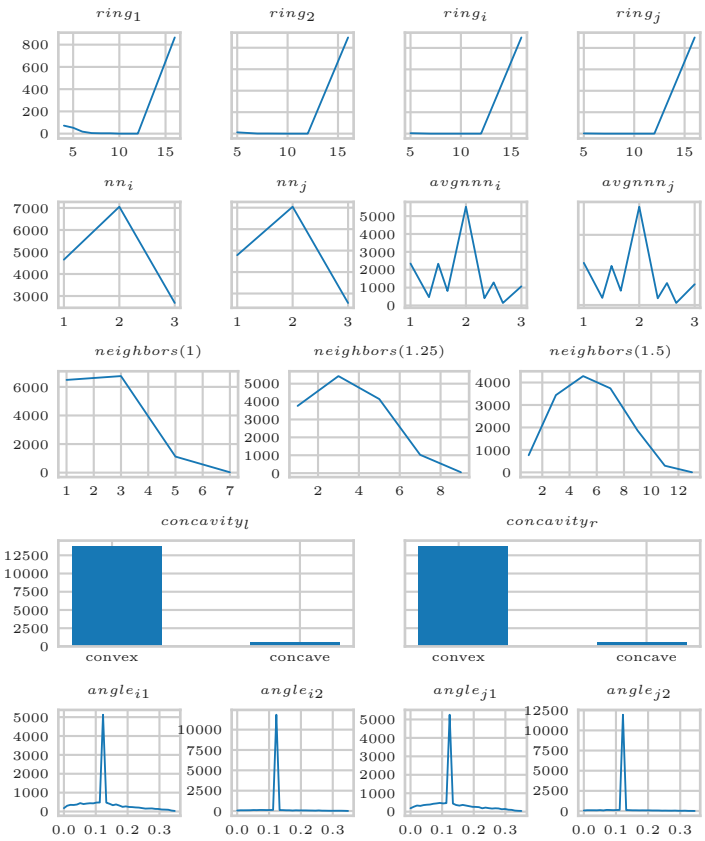


Figure 3.10: Distributions of the order parameters for $\rho\sigma^2 = 0.75$ $\beta\epsilon = 2$

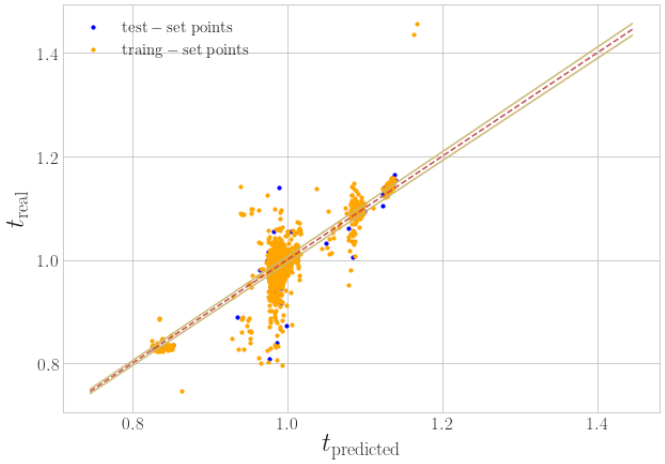


FIGURE 3.11: Result of the trained NN for $\rho\sigma^2 = 0.75$ and $\beta\epsilon = 2.0$. The dashed line shows the true bond breaking times. The green lines enclose an area around the target values corresponding to $\pm 2\sigma_{err}$, indicating the statistical uncertainty in the measured bond breaking times. The resulting accuracy of the prediction is $\mathcal{R} = 0.82$.

Here we can observe that the resulting $\mathcal{R} - correlation$ is pretty high when compared with other simulations, even though the result is not properly a straight line. Looking at the graph, it can be observed that the trend of the *true line* has been generally caught, but the predictions tend to be localized in separated areas, probably associated with clusters of particles with similar properties. In Fig. 3.12 the breaking times of each edge are coloured in such a way to emphasize how the prediction are distributed. The different kinds of edges are separated in 4 classes of prediction with very similar colours. These are the opened clusters, with a low breaking time, their ending edges, with a slightly longer one, and the closed rings together with the single-edge clusters, with the longest time to break. This result can be explained by saying that while trained to get a continuous output function, in some sense the NN learns how to *classify* this types of bonds, getting instead a quasi-discrete signal as result.

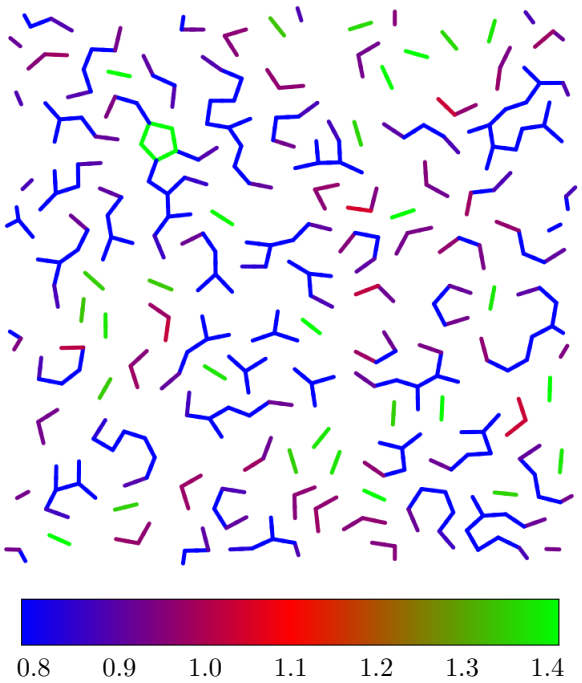


Figure 3.12: Representation of the predicted breaking time for a snapshot of the system taken at $\rho\sigma^2 = 0.75$ and $\beta = 2.0$.

In order to see which order parameters are more valuable in the training of the network, we tried to exclude some of them. The relevant result from this analysis is the fact that the parameters associated with the rings are of primary importance even if there is a very small number of them. When training the NN without these values we in fact obtain a dramatic decrease in the final accuracy. One reason for that is the fact that in this case the network as almost no way to distinguish between the particles involved in loops from the others. Then, even if the former are much less than the latter, the whole prediction is affected by this.

Finally, despite this anomalous classification feature the NN present, by looking at the representation of Fig. 3.13 it can be observed that the prediction is pretty accurate for most of the bonds. A more accurate comparison of the two values for each of

the bonds would give that the greatest error is associated with the big central ring. Such observation may suggest the fact that the training set does not contain enough rings to make the NN able to deal with them.

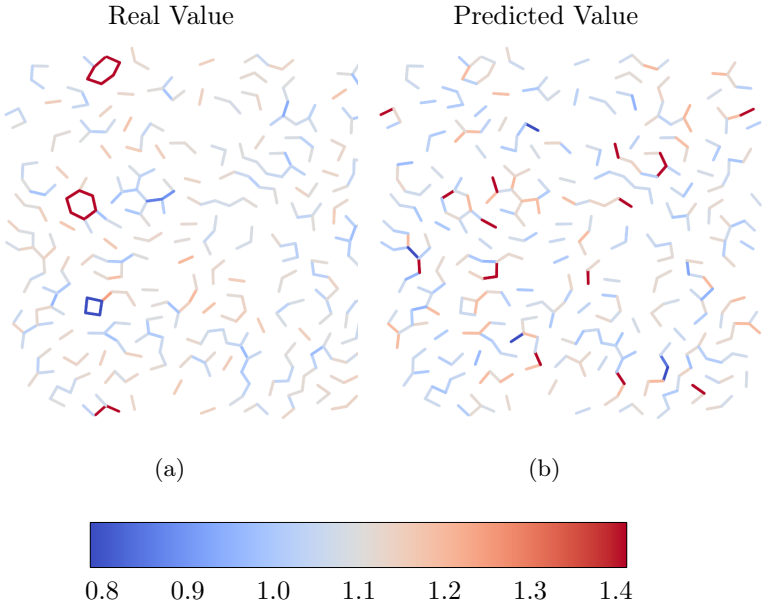


FIGURE 3.13: Comparison between the real and the predicted breaking times of the bonds in a snapshot taken at $\rho\sigma^2 = 0.75$ and $\beta\epsilon = 2.0$.

3.3.2 Low temperature

In the case of lower temperatures (here $\frac{k_b T}{\epsilon} = 0.2$), the system is already organized in one system-spanning cluster, where most of the particles have at least one connection with the others and there is a high distribution of rings. From Fig. 3.9b it is possible to observe the presence of many rings of small length, but also some rings much larger, in a network which is significantly more connected when compared to the previous one. This observation can suggest the fact that predicting this situation could be more interesting than the previous case. One more more hint enforcing

this is the fact that the values of the breaking times are much more spread out, meaning that some bonds break fast, but for others the local connectivity make this event much slower. Going deeper in the analysis of the data distribution (Fig. 3.14) is it possible to observe that the highest part of the rings is now concave. The reason is that all the big rings that are now present tent to be stretched in the network, resulting concave.

By training the network on this data we observe that it needs more iterations to reach a significant accuracy. This is one more observation confirming the fact that the system is now more complex to predict. Plotting the result in a graph displaying the real values over the predictions as before (Fig. 3.16) we observe that the accuracy is lower than in the previous case, but the predictions are spread in one single cluster. Despite the error in the predicted value is high if compared with the statistical error, the resulting direction of the distribution of point follows the *true line*. In fact by looking at one analyzed snapshot at Fig. 3.16, it can be observed that the predicted times (Fig 3.16b) are for most of the bonds pretty similar to the real values (Fig 3.16a), meaning that the result of the NN is promising also in this situation.

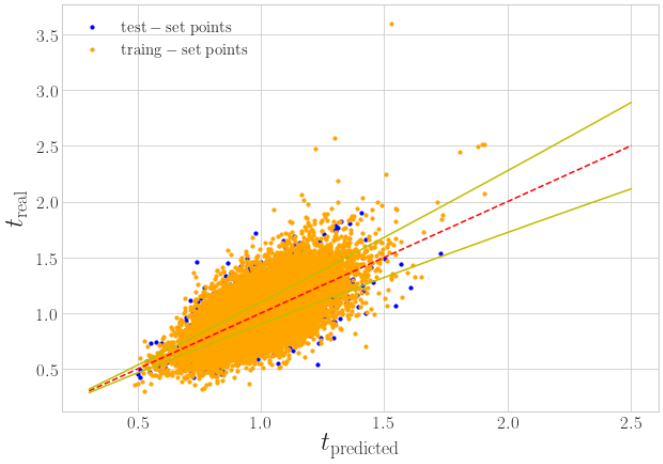


Figure 3.15: Result of the trained NN for $\rho\sigma^2 = 0.75$ and $\beta\epsilon = 5.0$. The resulting accuracy of the prediction is $\mathcal{R} = 0.62$.

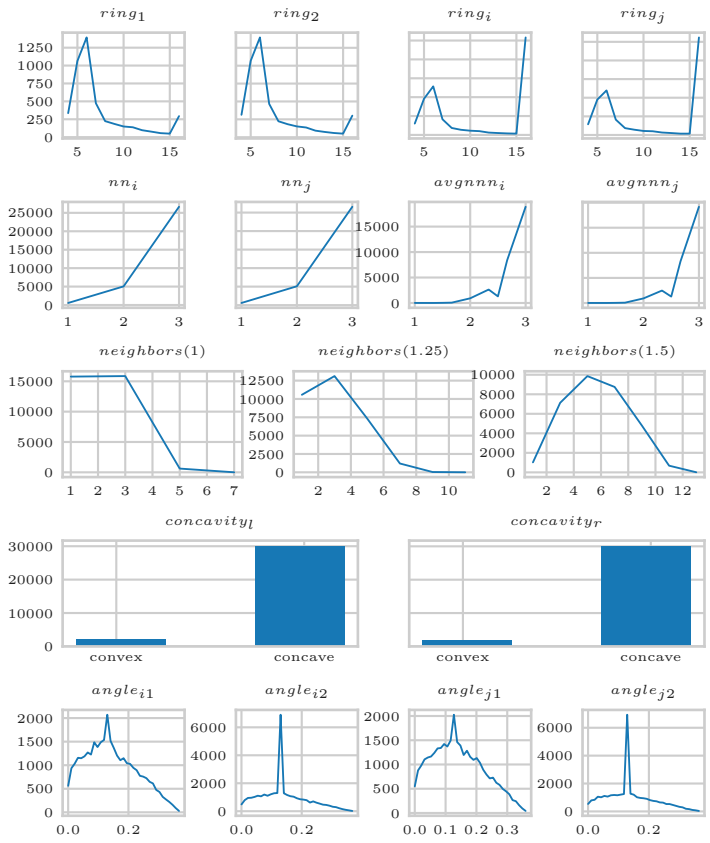


FIGURE 3.14: Distributions of the order parameters for $\rho\sigma^2 = 0.75$ $\beta\epsilon = 5$

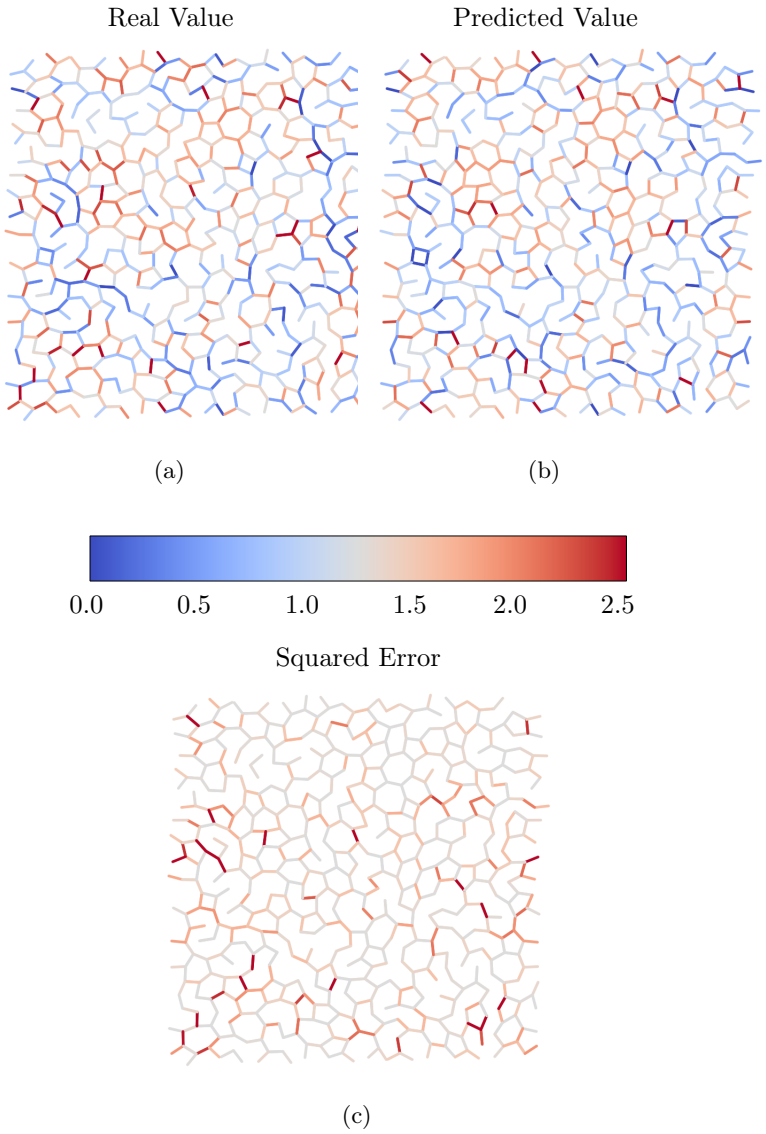


Figure 3.16: (a) Representation of the real breaking time of the bonds in a snapshot. (b) Predicted value of the breaking time. (c) Absolute error in the prediction, shown in arbitrary units.

Moving then to Fig. 3.16c, we color bonds based on the squared error between the predicted and the real result. Analyzing this graph it can be seen that most of the largest errors the NN commits come from small rings. One possible way to improve the result would be the one of incorporating some extra information on the rings by adding more complex order parameters. Anyway this figure does not emphasize any pattern, suggesting no particular way of constructing new input parameters. On the other hand, both tests done excluding some order parameters and modifications on the NN structure do not give any relevant improvement, suggesting that in some sense the predictive capacity of a MLP on this problem has been reached.

3.3.3 Overall view

In order to have a more organic look at the performances of a neural network to our system, we repeated the training procedure for a larger set of points and for different NN structures. Tuning the number of hidden layers and neurons per layer we built 4 different NN structures and considered for each point the one giving the best correlation. As previously stated, in all these comparisons we observed that the accuracy was very similar for all the NNs analyzed, confirming the fact that at this point the predictions are robust with respect to changes in the learning model complexity. A plot collecting the $\mathcal{R}-correlations$ for different temperatures at fixed density $\rho\sigma^2 = 0.75$ is shown in Fig. 3.17. We can observe that the predictive power of the network decreases as the temperature decreases. This can be explained by assuming that the breaking time variability increases as as the temperature decreases (*i.e.* the inverse temperature increases), making breaking times more and more difficult to predict. Another possible explanation is the possibility that at sufficiently low temperatures, bond breaking times become dependent on the local structure over a larger local region, which is not considered by our order parameters. This is a reasonable interpretation because, as in many physical systems, we expect that the correlation length tent to increase when the temperature goes down. One further step will be the one of taking these higher order correlations into account. One possibility to perform this task is the *Graph Network* (GN) method, which we will discuss in the next section.

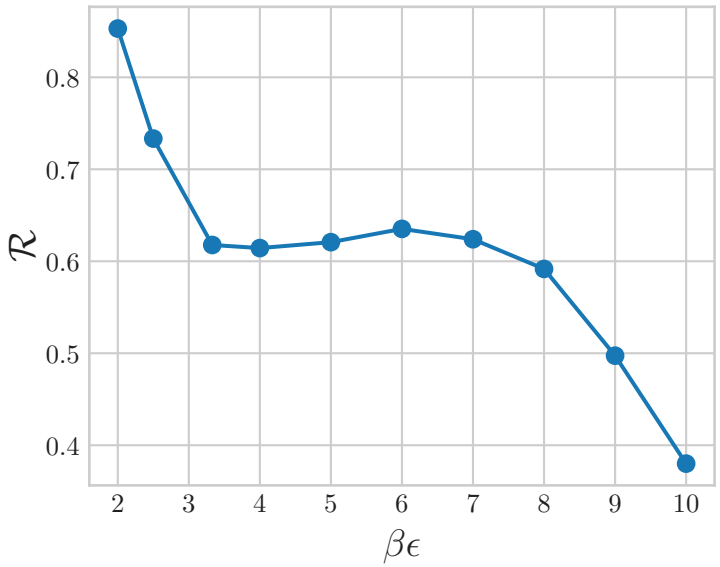


Figure 3.17: $\mathcal{R}-$*correlation* trend for predictions at density $\rho\sigma^2 = 0.75$ and variable temperature.

Summing up where we are, the predictions up to now aim to find the bond breaking time when the system is equilibrated at fixed density and temperature. We have proven that a NN is actually able to work with reasonably high accuracy in those situations by only considering the information of particle position and bonding pattern of single snapshots. This is an interesting result, but at this point nothing can be said about the possibility to use an algorithm of this kind to study the most interesting problems on glasses, such as glassy phase transition. In order to perform this task, a NN trained at a certain state point has to maintain a stable and sufficiently high accuracy in the predictions done even at different conditions. In this way it would be able to follow the system evolving at varying temperature and/or density. Starting from a network trained at each temperature, we then evaluated the $\mathcal{R}-$*correlation* it reaches when predicting from the

datasets of all the others. The result of this analysis is summarized in Fig. 3.18. Here we observe the fact that in some cases the trained network is more effective for temperatures different from the one it has been trained on (indicated in each curve with a larger spot). However, most of them effectively maintain a reasonably high correlation for a certain range temperatures. This suggests the idea that the structural information the algorithm learns from the snapshots is to some extent independent of the global parameters at which they are taken. Then the effect of the topology in the proximity of a bond is the same for different temperature within a certain range, and so one of the trained NNs can in principle follow the evolution of the system despite the fact that quantities such as temperature chance during the simulation.

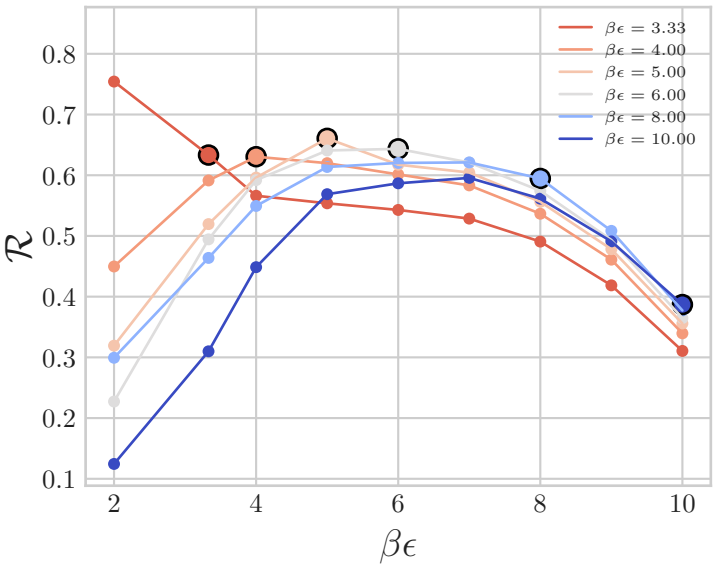


FIGURE 3.18: Correlation of NNs trained on snapshots taken at different temperatures and tested for some points in the range $\beta\epsilon = 2 \sim 10$.

3.4 Possible improvements

One possible way to improve the performance of the NN can be provided by the use of a slightly more complex structure, whose complexity can be inferred by the system itself, a GN [58]. This approach is relatively new in this field of research, but has already proven to be surprisingly effective for several machine learning problems [59; 60],including predictions for particle mobility in glassy materials [61].

The core idea of the method is to feed the algorithm with more information about the real system by constructing a NN that has a structure similar to the system under consideration, starting from the particles and the connection between them. Interpreting our system as a graph of nodes connected by bonds, it is sensible to assume that the output for any bond can be affected by information only related to nearby other bonds, or nearby particles. By constructing our NN to include so-called *graph layers*, adopting the same graph structure of the bonding graph of our system, we can explicitly incorporate this idea into our model.

At this point it is useful to recall that the object that we want to predict, which is the bond breaking time, is expected to be strongly related with the stresses in the network. In the previous approach, we were considering only local order parameters (such as cycles and neighbors positions). We can imagine that the NN extracts such non-local relations as an approximation from the local stresses to which it has access. Moreover, since the NN is fed with points from a data-set in which the edges are treated as independent, we were implicitly excluding all the information beyond the relation between the nearby ones, or at least we were again hoping the network would be able to learn how to extrapolate it from local data. Starting from this, it is not unreasonable to think that we can achieve better predictions by basing them also on the intermediate or final results (*i.e.* node outputs within the neural network) produced for the bonds close to the one we are considering.

In order to better visualize the advantages of a more careful approach, we can see the NN as a *random walker* (RW) moving its steps in the space of the possible solutions. It starts following totally random paths at the beginning, then the learning procedure

suggests him how to find the right way. This strategy, as previously said, gives sensible result as long as the RW is actually able to span a broad region of the space, then it is also required that the space is deep enough. This second condition means that a not broad enough point of view (*i.e.* a higher dimension of the space) easily lead to the existence of many equally good solutions, which could not be exactly equal if the algorithm has enough information to distinguish between them. In this framework, the extra information we are adding is seen as what in literature is called a *relational inductive bias* [58], which is in some sense a *hint* that allows the RW to prioritize some class of solutions with respect to others. The improvement is obviously given by the fact that a less generic network can focus on relevant solution much faster, and hopefully reach even better predictions.

Moving to the details of the method, we can assume to gather all the information of our system, collected into snapshots, in the following data structures:

$$\begin{aligned} D_{\text{in}} &= \{\mathbf{v}_i; \mathbf{e}_j\},\ i = 1, \dots, N_v; j = 1, \dots, N_e \\ G_{\text{in}} &= \{(s_j, t_j)\},\ j = 1, \dots, N_e, \end{aligned} \tag{3.20}$$

where N_v is the number of particles in the snapshot and N_e is the number of bonds. All the information of the order parameters associated with the edges is now stored into the vectors $\mathbf{e}_j$, with extra attributes associated with the vertices contained in the vectors $\mathbf{v}_i$. The particle order parameters we added are the number of nearest neighbors and their directions. The data structure G_{in} is then the graph representation of the snapshot made through the edge-list of the formed bonds. The bonds are stored as directed edges, meaning that in the edge (s_j, t_j) s_j and t_j are respectively source and target for that bond. Obviously for this problem bonds are symmetric relations, then in the edge-list both (s_j, t_j) and (t_j, s_j) have to be present. We want now to use these features to train a new network, incorporating the *graph idea* in a new *deep learning component*, as the LP and MLP previously used. This block, which we can call a GN, has to take as input both the attributes in D_{in} and the structural information in G_{in}, returning updated attributes in D_{out}. The difference is that each new feature is calculated using the same function, which only takes into account the features of edges or vertices that are from at most

one bond away. In a certain way of thinking, the improvement does not come the *presence* of a particular new object, but from the *absence* of something, namely some edges. In other words, the network now knows that each vertex directly depends only on its neighbors, being less and less sensible to the changes of the other vertices as they are far in term of connection from it. Moreover, in this way the (non-linear) function which calculates the updated features for each vertex is the same for each of them, independently on the graph size. This means that the complexity of the NN does not need to grow when considering a larger system of particles (which would correspond to a larger graph).

Assuming a GN as a *black-box*, that will be defined further on, we can construct a final NN as in Fig. 3.19. The two parts of D_{in} enters in the graph layers through two different *encoders*, which are MLPs. These are implemented in order to turn the number of order parameters to the desired dimension of the nodes and edges attributes. The subsequent Graph layer is then fed with the *pre-processed* features and with G_{in}, which remains the same for all the following layers. Information can travel further along the network, getting updated by all the chained GNs, which take the D_{out} of the previous layer as input attributed and return it modified for the one after.

3.4.1 Graph network implementation

In the framework of the GN layer proposed in Ref. [58], the graph that it operates on can be defined as the structure $G = (V, E)$, where V and E are respectively the set of vertices and edges of a usual graph. Both vertices and edges are also associated with one or more attributes, which is supposed to change during the learning process. Hence, we have $V = \{\mathbf{v}_i\}_{i=1:N_v}$, where N_v is the number of vertices and $\mathbf{v}_i$ is the attribute (a vector of values) related to the vertex i. In the same way $E = \{(\mathbf{e}_j, s_j, t_j)\}_{j=1:N_e}$, with N_e the number of edges and $\mathbf{e}_j$ the attribute of the edge j, which connects s_j with t_j. In our model, $\mathbf{e}_j$ is a set of values that are converted to a single parameter through the edge decoder, turning into the predicted breaking times of the network. The attributes for the vertices are instead parameters which do not appear in the final output, since they are not necessary in our study. Hence, they will be used in the learning process to carry

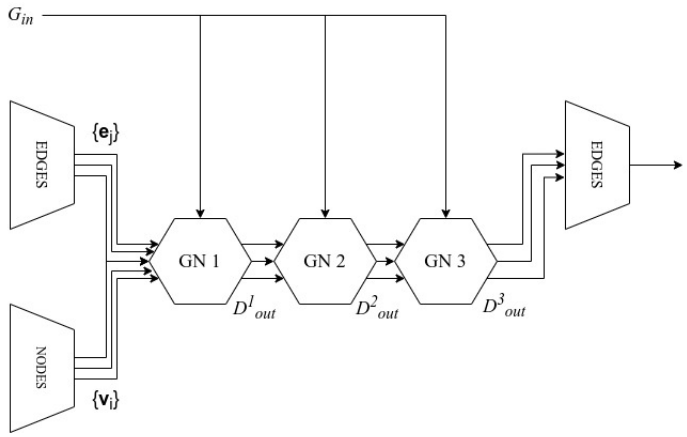


FIGURE 3.19: Diagrammatic representation of the full network. The hexagons represent the GN layers, while the trapezoids are the MLP encoders and decoder

information between edges but the output will be neglected at the end. This framework is fully general, meaning that in principle a GN can be defined on an *directed multi-graph*, in which the nodes i, j can be connected by any number of edges, each of them with a defined direction (*i.e.* $(i, j) \neq (j, i)$). In our system we have a symmetric bonding relation as the definition of the edges, so a connection between two particles is always encoded as two directed edges with opposite direction.

3.4.1.1 Computational Step

The execution of the network with this extra learning block is essentially similar to the one described in the previous section. We will then have repetition of steps of feedforward passage and a consequent backpropagation algorithm to update the weights. The difference lies in the way we update the block of information that passes through the network within the graph layers. For a given graph layer, we define $V' = \{\mathbf{v'}_i\}_{i=1:N_v}$ as the set of all the updated attributes of the vertices, $E'_i = \{(\mathbf{e'}_j, i, t_j)\}_{j=1:N_e, t_j \in \partial i}$ as the set of the updated attributes of the edges starting from the vertex i, and the set of all the updated attributes of the edges

as the union $E' = \bigcup_i E'_i$. It is then possible to introduce the relations:

$$\begin{aligned} \mathbf{e'}_j &= \phi^e(\mathbf{e}_j, \mathbf{v}_{s_k}, \mathbf{v}_{t_k}) \\ \mathbf{v'}_i &= \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i) \end{aligned} \tag{3.21}$$

$$\bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i).$$

The ϕ functions are the "update" relations for the attributes of the network which, as in a usual linear layer, would be the composition of a linear combinations and an activation functions. The last line of Eq. 3.21 instead introduces the "aggregation" function, which take a set as an input and give a reduced single element representing the *aggregated information*. It is important to observe, as mentioned before, that the updated attribute for the nodes only depend on the neighbors of each through the aggregated parameter $\bar{\mathbf{e}}'_i$. Since the update function must be the same for each node, regardless of the number of neighbors, the function $\rho^{e\to v}$ is used for condensing the vector of data into just one value. This can be done by designing this function to be invariant under permutations of its inputs. One logical choice will then be to return the sum of all the entering parameters. In general it can be constructed as any commutative relation of all the inputs, such as summations, means and and maximum values.

Starting from the current values of the attributes, at each step the GN starts updating through ϕ^e the edge ones, which only depends on the old parameters. The new values obtained in this way are collected in sets E'_i for each node, used as an input for the $\rho^{e\to v}$ functions to get the aggregated attributes for the neighbors. Then the node attributes can be updated by applying ϕ^v on its argument for each vertex. Finally the procedure outputs the updated attributes in V' and E'. These passages are summarized in algorithm 5. Note that this approach closely follows the method outlined in Ref. [58], but leaves out the possible effect of any global parameters.

3.4.1.2 Implementation details

One of the most natural ways to define the previously introduced functions is to treat them as a generalization of the usual LP

Algorithm 5 Update step for a GN block

function GN_STEP(E, V)
  for $j \in \{1..N_e\}$ **do**
    $\mathbf{e'}_j \leftarrow \phi^e(\mathbf{e}_j, \mathbf{v}_{s_k}, \mathbf{v}_{t_k})$ ▷ Updates edge attributes

  for $i \in \{1..N_v\}$ **do**
    $E'_i \leftarrow \{(\mathbf{e'}_j, i, t_j)\}_{j=1:N_e, t_j \in \partial i}$
    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}(E'_i)$ ▷ Aggregates edge attributes per node
    $\mathbf{v'}_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i)$ ▷ Updates node attributes

  $V' \leftarrow \{\mathbf{v'}_i\}_{i=1:N_v}$
  $E' \leftarrow \{(\mathbf{e'}_j, s_j, t_j)\}_{j=1:N_e}$
  return $(E', V', \mathbf{u}')$

update functions. This can be done constructing the ϕ functions as a standard MLPs, consisting of a sequence of linear NN layers and activation functions as described in the previous Section. An analogous reasoning can be applied to interpret this approach also to the edges attributes. Since the two update functions have a different number of input variables, they are in principle different, but both are MLPs taking in a single vector of inputs:

$$\begin{aligned} \phi^e(\mathbf{e}_j, \mathbf{v}_{s_k}, \mathbf{v}_{t_k}) &:= MLP_e([\mathbf{e}_j, \mathbf{v}_{s_k}, \mathbf{v}_{t_k}]) \\ \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i) &:= MLP_v([\bar{\mathbf{e}}'_i, \mathbf{v}_i]). \end{aligned} \tag{3.22}$$

In the previous relation, the symbol $[\cdot]$ indicates a vector concatenating all the attributes inside, as they are the signals arriving from the previous layer of the GN, or (for the first graph layer) from the encoder.

On the other hand, any symmetric relation of the entries can be used to define the aggregation function. In order to keep both information on the number of the neighbors for each input (indicated as n_i for the node i) and an average of the features each of them carries, we constructed $\rho^{e\rightarrow v}(E'_i)$ as a concatenation of the summation of all the inputs and its mean value, normalized to the number of neighbors n_i. The resulting aggregation function is

the following:

$$\rho^{e\to v}(E_i') := \left[\sum_{\{k:k\in\partial i\}} \mathbf{e'}_k;\ \frac{1}{n_i} \sum_{\{k:k\in\partial i\}} \mathbf{e'}_k \right] \tag{3.23}$$

A GN constructed in such way can be thought as a black-box accepting input signals, processing them returning some output with a well-defined update procedure. The most important feature here as that each GN layer only updates the local edge and node properties based on their immediate surroundings. However, a series of such layers allow variations in local structure progressively further away to affect the final result for any given edge in the systems. All these ingredients are enough to make it usable as a building block for the wider NN.

3.4.2 Results

Training the new network structure on the same data we observe significant improvements for low temperatures, as shown in Fig. 3.20. This is a confirmation of the idea that introducing more information on the correlation between bonds over a larger region, as the GN implementation does, is effective in this regime. For relatively high temperatures the GN does not bring improvement, and instead it gets even worse results when compared with the MLP. This can be explained by the fact that at these temperatures it is sufficient to look at the correlations only with the order parameters defined in the previous sections. In this way, the increase in complexity introduced by the use of the graph layers can not be justified. Moreover the snapshots at these state points induce highly disconnected topologies, which can result in a drop down of the performances. Similar to the results from the MLP predictions we observe a decrease in the correlation at very high temperatures ($\beta\epsilon > 8$). This may be attributable to the lower quality of the data at these low temperatures, where the number of bond-breaking events observed in a given simulation is relatively low.

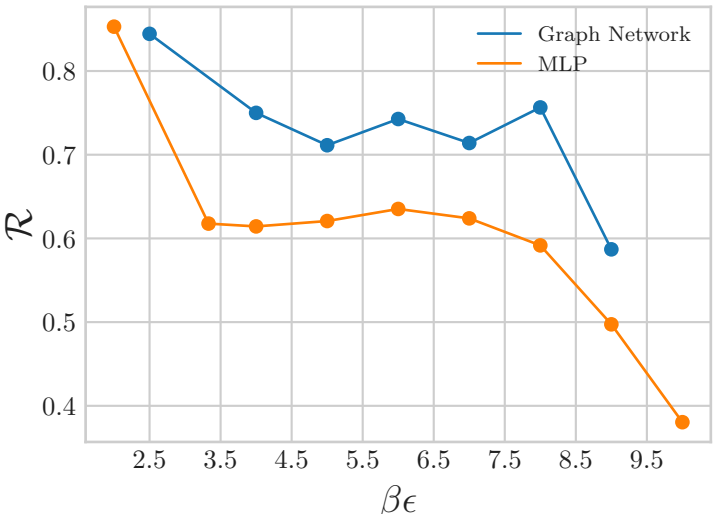


FIGURE 3.20: Comparison between the $\mathcal{R} - correlation$ trend of the GN and the and the MLP for predictions at density $\rho\sigma^2 = 0.75$ and various temperatures.

As we did for the MLP, the trained GN can be tested on datasets collected at variable temperatures, in order to verify its capability to analyze eventual out of equilibrium simulations. In Fig. 3.21 are gathered some of the performance trend for different starting temperatures. Looking at the curves, we observe that the ones at sufficiently low temperature present a plateau in the proximity of the state point of the training, confirming the fact that also this method can be effective for this purpose. Comparing the overall performances of MLP, and with even better results the GN approach, we are finally able to say that both methods can catch the features behind the bond breaking events, with a certain degree of robustness to changes in the conditions of the system.

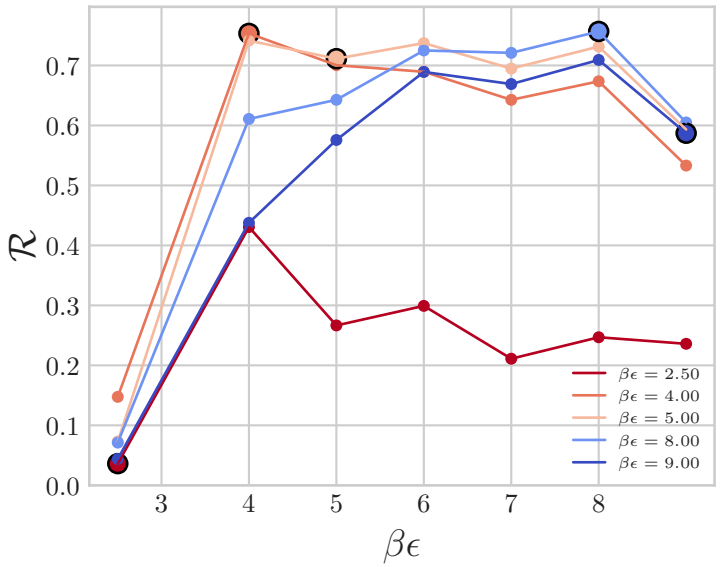


Figure 3.21: Correlation of GNs trained on snapshots taken at different temperatures and tested for some points in the range $\beta\epsilon = 2 \sim 9$.

Conclusions 4

We now come back to our *movie observer* from the introduction, who speculated on the possibility of reconstructing the full time evolution of a glassy system only starting from a single snapshot. In the previous chapter we effectively see how this idea can be concretely implemented through the use of NN. Applications of this technique are widespread in a huge amount of modern research environments. We can in fact predict information about a physical system, a social community, a human speech, and in principle an infinite number of other fields [62; 63; 64]. The only limit is to find a smart way to translate our perception from the world into a quantitative model and then in a set of parameters that a *universal black-box* can read and base its answer on. This is obviously not completely true. In fact, while not considering the difficulties underlying the definition of a really representative model of our problem, we saw how even this *black-box* should be *specialized* in order to give sensible results.

In the previous chapter we saw that a classical MLP applied to our patchy-particle network can get reasonably good predictions, but every configuration of neurons and layers seems to present a sort of upper bound in the reachable accuracy. The improvement we adopted was to consider the data in a different way, incorporating the global network topology by using a GN. This effectively

led to a gain in the network performance, suggesting the possibility that the information contained in the snapshots is sufficient to perform the task, but the local way we were looking at them was not sufficient to catch all the features. From another point of view, rather then modifying the way of considering the data, what we did was change the NN structure in order to incorporate in it the topological information of the real system. The resulting network is obviously less general, in the sense that this structure is not guaranteed to have better performances also when looking at other problems, but for this reason it is in some sense more specialized to execute its actual job. In particular, the GN structure allows the NN to also consider features of the system further away from the particle or bond under consideration, improving its predictive power.

This interpretation does some damage to the idea that a NN can be a *universal solver* for all problems, preferring instead a different network structure to solve different problems. Nevertheless looking from a broader level, machine learning is for sure an extremely powerful instrument to get to solve almost any problem, as long as we consider it from the right perspective.

The results we got from our implementations reflect one other relevant fact: even by constructing a NN which is specific for our problem, the error in the predictions are still not negligible for some configuration. Moreover this choice had the drawback of considerably increase the complexity of the predictive algorithm. Obviously we have no guarantee that our algorithm is the best possible one to predict bond breaking times for patchy particles. In principle there may be another method capable of perfectly performing this task. While admitting the existence of such *perfect algorithm*, which we do not have right now, in would be useful to look again with more attention to the main question of this thesis: *is the information contained in a single snapshot sufficient to predict the system evolution?*

Many studies shows that the local structure allow to predict in which regions rearrangement events are more likely to occur [65; 66], but the question is actually more profound. Our tendency to lean towards a positive answer increased when we incorporate in our learning model the global topological information as a *relational inductive bias* [58]. This could suggest us to move a step back and ask ourselves which is the role of such biases, or

in other way of some *a priori information*, in making predictions on our or any system. One first answer to this arrives from what theoreticians call the *Ugly Duckling theorem* [67]. This theorem, developed for classification problems but which can be conceptually extended also to regressions, asks whether it is possible to make classifications without any sort of bias. It takes its name form the story of *The Ugly Duckling* from H. C. Andersen, which effectively shows that a duckling may be as similar to a swan as two duckling are between each others. The argument says that without any prior information about the classification we have to perform (*i. e.* without knowing if we have to distinguish between a duckling and a swan) it is impossible to build a classifier, because in some way all the possible classification are equivalent.

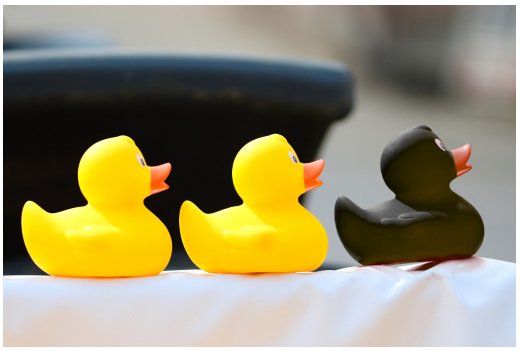


FIGURE 4.1: A duckling can be as similar to a swan as to another duckling.

Coming back to our problem, in all the supervised learning algorithms we built, useful information to distinguish between different configuration was given by the supervised training procedure. Rather that asking if our NN is capable of recognising or not a *swan*, this reasoning is useful to stress out one more time the fact that a good choice of the information given to the algorithm is of primary importance in determining its performances. Then if a *Universal algorithm* is impossible, it is reasonable to assume that even a *perfect algorithm*, able to predict in the same way all the features of a system as the one we are studying, is likewise unachievable. Then one further step would be the one

of optimizing an algorithm in order to find the best compromise between generality for our purposes and accuracy.

In this thesis the NNs were trained in order to predict which bonds are more likely to break from a certain configuration, following the idea of being able to determine the evolution of the system. Going further in this direction, the next step would be the one of repeating an analogous reasoning to the bond formation. Looking at a fixed snapshot, it is straightforward to state which are the pairs of particles to analyze when looking for possible breaking events, because they can only be chosen among the connected ones in the snapshot topology. Less direct but still doable would be to predict the most likely formed bonds, which can instead originate from every particle with the unbonded patches of all the others. Moreover, in a 2D system where the bond network percolates, bonds can only form between two free patches that are both on the inside of a closed loop, which drastically restricts the possibilities. What we learned from glasses, and in particular network glasses, is that their dynamics are dominated by rare rearrangement events involving the bonds with long time-scale compared with the fast movement induced by thermal fluctuation. As said in the previous chapters, this separation of time scales makes studying the full evolution of such materials difficult in experiments. This issue is partially solved by computer simulations, but is still problematic for very low temperatures, because the gap between the two time-scales grows more and more as the temperature decreases. As a consequence, even the equilibration of the system at very low temperature becomes practically unfeasible, setting a limit to the power of usual simulations, and then to our comprehension of these materials. A reliable method of predicting these events can then become a key feature of a new class of *machine learning-based simulation algorithms*, which can overcome this barrier.

As we saw, our NN trained at one state point is capable of making predictions even for different ones. At this stage we only used this feature to study situations which could be analyzed also with usual methods, but in principle NNs could even be used to predict the dynamics of networks at temperatures too much low to sample in conventional simulations. Then these results lay the foundation for this new way of simulating network glasses, which can bring our understanding of these materials to a next level,

and maybe lead to a new frontier of their application to everyday life.

Bibliography

[1] A. B. Pawar and I. Kretzschmar, *Fabrication, assembly, and application of patchy particles*, Macromolecular rapid communications **31**, 150 (2010).

[2] E. Locatelli, P. H. Handle, C. N. Likos, F. Sciortino, and L. Rovigatti, *Condensation and demixing in solutions of dna nanostars and their mixtures*, ACS Nano **11**, 2094 (2017). PMID: 28157331.

[3] L. Berthier and G. Biroli, *Theoretical perspective on the glass transition and amorphous materials*, Rev. Mod. Phys. **83**, 587 (2011).

[4] G. L. Hunter and E. R. Weeks, *The physics of the colloidal glass transition*, Reports on progress in physics **75**, 066501 (2012).

[5] F. H. Stillinger and P. G. Debenedetti, *Glass transition thermodynamics and kinetics*, Annual Review of Condensed Matter Physics **4**, 263 (2013).

[6] C. P. Royall and S. R. Williams, *The role of local structure in dynamical arrest*, Physics Reports **560**, 1 (2015).

[7] M. Micoulaut, *Relaxation and physical aging in network glasses: a review*, Reports on Progress in Physics **79**, 066504 (2016).

[8] M. T. Cicerone and M. Ediger, *Relaxation of spatially heterogeneous dynamic domains in supercooled ortho-terphenyl*, The Journal of chemical physics **103**, 5684 (1995).

[9] L. Berthier, G. Biroli, J.-P. Bouchaud, L. Cipelletti, D. El Masri, D. L'Hôte, F. Ladieu, and M. Pierno, *Direct experimental evidence of a growing length scale accompanying the glass transition*, Science **310**, 1797 (2005).

[10] P. Wolynes and V. Lubchenko, *Structural Glasses and Supercooled Liquids: Theory, Experiment, and Applications* (Wiley, 2012).

[11] A. Einstein. *Investigations on the theory of brownian motion. translated by ad cowper*, (1905).

[12] E. Bianchi, J. Largo, P. Tartaglia, E. Zaccarelli, and F. Sciortino, *Phase diagram of patchy colloids: Towards empty liquids*, Physical review letters **97**, 168301 (2006).

[13] F. Sciortino and E. Zaccarelli, *Reversible gels of patchy particles*, Current Opinion in Solid State and Materials Science **15**, 246 (2011).

[14] J. Russo, J. Tavares, P. Teixeira, M. T. da Gama, and F. Sciortino, *Re-entrant phase behaviour of network fluids: A patchy particle model with temperature-dependent valence*, The Journal of chemical physics **135**, 034501 (2011).

[15] W. R. Smith and I. Nezbeda, *A simple model for associated fluids*, The Journal of chemical physics **81**, 3694 (1984).

[16] M. Wertheim, *Fluids with highly directional attractive forces. i. statistical thermodynamics*, Journal of Statistical Physics (1984).

[17] M. Wertheim, *Fluids with highly directional attractive forces. iv. equilibrium polymerization*, Journal of Statistical Physics (1986).

[18] I. Nezbeda, J. Kolafa, and Y. V. Kalyuzhnyi, *Primitive model of water: Ii. theoretical results for the structure and thermodynamic properties*, Molecular Physics **68**, 143 (1989).

[19] M. H. Ford, S. M. Auerbach, and P. Monson, *On the mechanical properties and phase behavior of silica: A simple model based on low coordination and strong association*, The Journal of chemical physics **121**, 8415 (2004).

[20] J. E. Shelby, *Introduction to glass science and technology* (Royal Society of Chemistry, 2005).

[21] R. P. Sear, *Phase behavior of a simple model of globular proteins*, The Journal of chemical physics **111**, 4800 (1999).

[22] J. P. Doye, A. A. Louis, I.-C. Lin, L. R. Allen, E. G. Noya, A. W. Wilber, H. C. Kok, and R. Lyus, *Controlling crystallization and its absence: proteins, colloids and patchy models*, Physical Chemistry Chemical Physics **9**, 2197 (2007).

[23] N. Kern and D. Frenkel, *Fluid–fluid coexistence in colloidal systems with short-ranged strongly directional attraction*, The Journal of Chemical Physics **118**, 9882 (2003).

[24] E. Bianchi, P. Tartaglia, E. Zaccarelli, and F. Sciortino, *Theoretical and numerical study of the phase diagram of patchy colloids: Ordered and disordered patch arrangements*, The Journal of chemical physics **128**, 144504 (2008).

[25] N. Dorsaz, L. Filion, F. Smallenburg, and D. Frenkel, *Spiers memorial lecture: Effect of interaction specificity on the phase behaviour of patchy particles*, Faraday discussions **159**, 9 (2012).

[26] F. Sciortino, E. Bianchi, J. F. Douglas, and P. Tartaglia, *Self-assembly of patchy particles into polymer chains: A parameter-free comparison between wertheim theory and monte carlo simulation*, The Journal of chemical physics **126**, 194903 (2007).

[27] E. Bianchi, J. Largo, P. Tartaglia, E. Zaccarelli, and F. Sciortino, *Phase diagram of patchy colloids: Towards empty liquids*, Phys. Rev. Lett. **97**, 168301 (2006).

[28] F. Smallenburg, L. Leibler, and F. Sciortino, *Patchy particle model for vitrimers*, Physical review letters **111**, 188002 (2013).

[29] Q. Chen, S. C. Bae, and S. Granick, *Directed self-assembly of a colloidal kagome lattice*, Nature **469**, 381 (2011).

[30] S. Biffi, R. Cerbino, F. Bomboi, E. M. Paraboschi, R. Asselta, F. Sciortino, and T. Bellini, *Phase behavior and critical activated dynamics of limited-valence dna nanostars*, Proceedings of the National Academy of Sciences **110**, 15633 (2013).

[31] I. Saika-Voivod, F. Smallenburg, and F. Sciortino, *Understanding tetrahedral liquids through patchy colloids*, The Journal of chemical physics **139**, 234901 (2013).

[32] H. Tong and H. Tanaka, *Structural order as a genuine control parameter of dynamics in simple glass formers*, Nature Communications **10**, 1 (2019).

[33] S. Marín-Aguilar, H. H. Wensink, G. Foffi, and F. Smallenburg, *Slowing down supercooled liquids by manipulating their local structure*, Soft Matter **15**, 9886 (2019).

[34] E. Boattini, S. Marín-Aguilar, S. Mitra, G. Foffi, F. Smallenburg, and L. Filion, *Autonomously revealing hidden local structures in supercooled liquids*, (2020).

[35] J. Paret, R. L. Jack, and D. Coslovich, *Assessing the structural heterogeneity of supercooled liquids through community inference*, The Journal of chemical physics **152**, 144502 (2020).

[36] R. Fantoni and G. Pastore, *Wertheim perturbation theory: Thermodynamics and structure of patchy colloids*, Molecular Physics (2015).

[37] A. Santos, S. B. Yuste, and M. López de Haro, *Contact values of the radial distribution functions of additive hard-sphere mixtures in d dimensions: A new proposal*, The Journal of Chemical Physics **117**, 5785–5793 (2002).

[38] A. Santos, M. DEHARO, and S. YUSTE, *An accurate and simple equation of state for hard disks*, JOURNAL OF CHEMICAL PHYSICS **103**, 4622 (1995).

[39] D. C. Rapaport, *The event-driven approach to n-body simulation*, Progress of Theoretical Physics Supplement **178**, 5 (2009).

[40] D. Rapaport, *The event scheduling problem in molecular dynamic simulation*, Journal of Computational Physics **34**, 184 (1980).

[41] L. Hernández de la Peña, R. van Zon, J. Schofield, and S. B. Opps, *Discontinuous molecular dynamics for semiflexible and rigid bodies*, The Journal of chemical physics **126**, 074105 (2007).

[42] F. Smallenburg and F. Sciortino, *Liquids more stable than crystals in particles with limited valence and flexible bonds*, Nature Physics **9**, 554 (2013).

[43] A. Ladd and L. Woodcock, *Triple-point coexistence properties of the lennard-jones system*, Chemical Physics Letters **51**, 155 (1977).

[44] B. J. Alder and T. E. Wainwright, *Phase transition for a hard sphere system*, The Journal of chemical physics **27**, 1208 (1957).

[45] D. Frenkel and B. Smit, *Understanding molecular simulation: from algorithms to applications* (Elsevier, 2001).

[46] A. Z. Panagiotopoulos, N. Quirke, M. Stapleton, and D. Tildesley, *Phase equilibria by simulation in the gibbs ensemble: alternative derivation, generalization and application to mixture and membrane equilibria*, Molecular Physics **63**, 527 (1988).

[47] A. Z. Panagiotopoulos, *Direct determination of fluid phase equilibria by simulation in the gibbs ensemble: a review*, Molecular simulation **9**, 1 (1992).

[48] D. G. S. Richard O. Duda, Peter E. Hart, *Pattern classification* (Wiley, 2001).

[49] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems **2**, 303 (1989).

[50] W. Sharpe, M.-Y. Chow, S. Briggs, and L. Windingland, *A methodology using fuzzy logic to optimize feedforward artificial*

neural network configurations, IEEE transactions on systems, man, and cybernetics **24**, 760 (1994).

[51] C. A. Williams, *Genetically evolving optimal neural networks*, (2005).

[52] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural networks **61**, 85 (2015).

[53] S. Ruder, *An overview of gradient descent optimization algorithms*, CoRR **abs/1609.04747**, (2016).

[54] C. Darken, J. Chang, J. Moody, et al. *Learning rate schedules for faster stochastic gradient search.* in *Neural networks for signal processing*, volume 2. Citeseer, (1992).

[55] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014).

[56] *Pytorch*, (2016).

[57] V. Subramanian, *Deep Learning with PyTorch* (Packt, 2018).

[58] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., *Relational inductive biases, deep learning, and graph networks*, arXiv preprint arXiv:1806.01261 (2018).

[59] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. *Interaction networks for learning about objects, relations and physics.* in *Advances in neural information processing systems*, pages 4502–4510, (2016).

[60] T. Wang, R. Liao, J. Ba, and S. Fidler, *Nervenet: Learning structured policy with graph neural networks*, (2018).

[61] V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. Schoenholz, A. Obika, A. Nelson, T. Back, D. Hassabis, et al., *Unveiling the predictive power of static structure in glassy systems*, Nature Physics **16**, 448 (2020).

[62] E. Guven. *System and method for classification of emotion in human speech*, (2013). US Patent App. 13/858,578.

[63] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti, *Tiles: an online algorithm for community discovery in dynamic social networks*, Machine Learning **106**, 1213 (2017).

[64] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, *Machine learning and the physical sciences*, Reviews of Modern Physics **91**, 045002 (2019).

[65] S. S. Schoenholz, E. D. Cubuk, D. M. Sussman, E. Kaxiras, and A. J. Liu, *A structural approach to relaxation in glassy liquids*, Nature Physics **12**, 469 (2016).

[66] 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*, Physical review letters **114**, 108001 (2015).

[67] S. Watanabe, *Knowing and guessing: a quantitative study of inference and information* (Wiley, 1969).

Index