ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS
DEPARTMENT OF DIGITAL SYSTEMS

NCSR DEMOKRITOS
INSTITUTE OF INFORMATICS AND
TELECOMMUNICATIONS

Investigation of Machine Learning-based Schemes for the Development of Coarse-Grained Force Fields for Organic Molecules

by

Spilios Dellis

Submitted

in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

September 2023

University of Piraeus, NCSR “Demokritos”. All rights reserved.

Author . . . . . .

II-MSc “Artificial Intelligence”
September 00, 2023

Certified by. . . . . .

George Giannakopoulos, Ph.D.
Functional Scientific Personnel (B)
Thesis Supervisor

Certified by. . . . . .

Niki Vergadou, Ph.D
Researcher (C)
Member of Examination Committee

Certified by. . . . . .

Theodoros Giannakopoulos, Ph.D.
Researcher (B)
Member of Examination Committee

Investigation of Machine Learning-based Schemes for the Development of Coarse-Grained Force Fields for Organic Molecules

By

Spilios Dellis

Submitted to the MSc "Artificial Intelligence" on
September, 2023,
in partial fulfillment of the
requirements for the MSc degree

Abstract

This master thesis presents the development of an innovative evaluation and ranking protocol that integrates physical insights to assess the performance of models and employs statistical tests for validation. The protocol was applied to address a multi-objective optimization problem related to self-adapting the weights of a Graph Convolutional Neural Network model. Specifically, the Graph Convolutional Neural Network model was designed to simulate a force field for predicting molecular system configurations in a coarse-grained setting. The SchNet architecture, a deep learning framework tailored for atomistic systems and capable of modelling quantum interactions in molecules, served as the foundation for the proposed approach. The training dataset consisted of multiple frames obtained from atomistic simulations of benzene molecules. This research mainly focused on achieving a self-balancing of weights within the dual components of the Graph Convolutional Neural Network model's loss function. This self-balancing aimed to optimize the trade-off between different objectives in the multi-objective optimization problem, enhancing the model's performance. To identify the most effective self-balancing method among various alternatives, the developed protocol was employed to evaluate the performance of each approach. The evaluation results revealed the most optimal self-balancing approach for this specific multi-objective optimization problem within the context of simulating a force field for predicting coarce-grained molecular system configurations. The presented methodology offers a valuable contribution to the field of deep learning applied to atomistic systems and multi-objective optimization, paving the way for further advancements in molecular dynamics simulations and related research.

Thesis Supervisor: Dr. George Giannakopoulos
Title: Investigation of Machine Learning-based Schemes for the Development of Coarse-Grained Force Fields for Organic Molecules

Acknowledgments

I wish to extend my sincere gratitude to the many individuals whose steadfast support and unwavering encouragement have been instrumental in bringing this master's thesis to fruition.

Foremost, I am deeply appreciative of the exceptional guidance and mentorship provided by my esteemed supervisor, Dr. George Giannakopoulos. His generosity in affording me the opportunity to immerse myself in such an exhilarating field and sharing his profound knowledge has been an enriching experience. Throughout the entirety of my research journey, his guidance and supervision were irreplaceable, contributing immeasurably to the quality of this work.

I extend my profound appreciation to the distinguished members of my advisory committee, Dr. Niki Vergadou and Dr. Theodoros Giannakopoulos. Their invaluable insights and constructive contributions have elevated the calibre of this thesis, leaving an indelible mark on its development.

I am indebted to Dr. Eleonora Ricci and Mr. Dimitris Gerakinis for their unwavering collaborative spirit, mentorship, and scholarly camaraderie. Their intellectual engagement has made this academic expedition intellectually invigorating and deeply gratifying.

To my family, I owe a debt of gratitude for their unwavering support, unwavering belief in my abilities, and the sacrifices they have made to ensure my success. Their enduring faith in my pursuits has served as an enduring wellspring of motivation and determination.

I am profoundly thankful to Katerina for her extraordinary patience, understanding, and unwavering encouragement, particularly in light of the unconventional decision to pursue an MSc degree at this stage of my life.

Lastly, I would like to express my appreciation to Nasia because she simply is.

To all those who have played a role, regardless of its magnitude, in this academic endeavour, your support, encouragement, and steadfast belief in my capabilities have been indispensable. I extend my profound gratitude for your contributions to the successful culmination of this master's thesis.

To my father...

Contents

List of Figures

List of Tables

1 Introduction

Molecular Dynamics (MD) has emerged as a powerful tool for simulating atomistic systems, providing researchers with an intricate understanding of materials properties at a granular level [1]. This computational technique bridges the gap between macroscopic properties and their microscopic origins, enabling the exploration of various phenomena that may be challenging or even impossible to observe experimentally [2]. By applying the principles of classical mechanics to atomistic and molecular systems, MD simulations can model complex material behaviours across a range of time scales [3]. The detailed data generated through these simulations has made them indispensable in numerous scientific disciplines, including materials science [4], biology [5], and physics [6].

Nonetheless, MD simulations come with a set of inherent challenges. The accuracy and reliability of these simulations greatly depend on the construction of interatomic potentials or force fields, which govern the interactions between atoms and molecules within the system [7]. The formulation of these potentials, a process known as parametrization, can be quite intricate. Additionally, the computational demands associated with these simulations often limit the temporal and spatial scales that can be probed [8].

Machine Learning (ML) has been identified as a potent tool for overcoming these challenges [9]. Within the realm of MD, ML algorithms have been harnessed to predict force fields from atomistic configurations, which significantly cuts down the computational costs associated with these simulations [10]. Various ML models, including Gaussian process regression, artificial neural networks, and deep learning models, have been adopted for this purpose, each bringing its unique strengths and limitations to the table [11].

Nevertheless, the integration of ML in MD simulations is not without difficulties. A key issue is the management of multi-component loss functions. These functions encompass multiple terms, each representing a different physical property or constraint that the model must maintain [12]. When these terms vary greatly in scale or significance, an imbalance may arise, potentially skewing the learning process. The model may become overly biased towards one component of the loss, at the expense of others, consequently diminishing its overall performance [13].

This problem has been encounter in several problems that have been addressed with ML and has led to the evolution of self-balancing methods [14]. These strategies seek to balance the contribution from each component to the total loss, thereby facilitating holistic learning. Techniques range from dynamically adjusting the weights of loss components during training [15, 16], to applying transformations that bring the components of loss to a similar scale [17, 18].

Evaluating the results of MD simulations involves a range of methodologies. A typical approach involves comparing the simulation results with experimental data [19]. This comparison offers a means to validate the accuracy and reliability of the simulation. Moreover, statistical measures such as the Root-Mean-Square Deviation (RMSD), Radial Distribution Functions (RDF), and Mean Square Displacement (MSD) can offer further insights into the simulation's quality [1].

Choosing the most appropriate self-balancing approach usually depends on the specific problem at hand. This selection process involves gauging the performance of different approaches based on various metrics, such as the accuracy of predictions and computational efficiency [20]. More advanced methods utilize statistical testing or optimization algorithms to ascertain the optimal approach [21].

The focal point of this master's thesis is to address the challenge of determining optimal weights for the components of a multi-objective loss function in ML model training. We specif-

ically focus on implementing self-balancing methods for the loss function components within the framework of a Graph Convolutional Neural Network (GCNN) scheme that has been previously developed for the description of intermolecular interactions. This scheme will be utilized for the molecular simulation of a bulk benzene liquid system at a coarse-grained level, using a single bead representation based on atomistic MD simulation ground truth.

The primary motivation behind this research stems from the importance of generating accurate and efficient force field simulations for molecular dynamics, towards which ML-based schemes could make significant contributions. The performance of GCNN models in simulating force fields is heavily influenced by the choice of weights assigned to the loss function components. As a result, there is a pressing need for self-adaptation methods that can effectively balance these components, thereby optimizing the performance of the GCNN model.

This master's thesis research aims to investigate the use various self-adaptation approaches of loss component weights in the GCNN model and establish a robust and consistent evaluation process for various self-adaptation approaches of loss component weights in the GCNN model. We implemented and compared several self-adaptation methods based on physical parameters and training efficiency metrics to achieve this.

The structure of this master's thesis unfolds as follows: Chapter 2 provides an introduction to the fundamental theories of Molecular Simulation (MS) and ML, along with an overview of the relevant scientific literature. Chapter 3 details the research methodology, outlining the adopted approach. Chapter 4 presents the experiments conducted and the corresponding results obtained. Finally, Chapter 5 offers a comprehensive discussion of the conclusions drawn from the research findings, alongside an exploration of any unresolved issues or open questions warranting further investigation.

2 Background Knowledge and Related Work

2.1 Molecular simulation

MS methods are based on the fundamental principles of statistical mechanics and provide a powerful means for the fundamental understanding of materials and the elucidation of the microscopic mechanisms that underly their macroscopic behaviour [1]. They enable the prediction of a wide range of properties and the unravelling of structure-performance relationships, contributing to the design of new advanced materials with controlled properties [22]. Molecular simulations involve a number of computational techniques that investigate the behaviour of materials at atomic and molecular levels in various lengths and timescales (Figure 1). This field has seen considerable expansion in recent years, largely attributed to advancements in computer hardware, software, the development of novel and more efficient scientific methods, and the increasing necessity to comprehend complex molecular systems [2, 22, 23]. Applications of molecular simulations span involve diverse fields of scientific and technological domains such as chemistry, physics, materials science, biology, biochemistry, nanotechnology, environmental and biomedical engineering and many others [24].

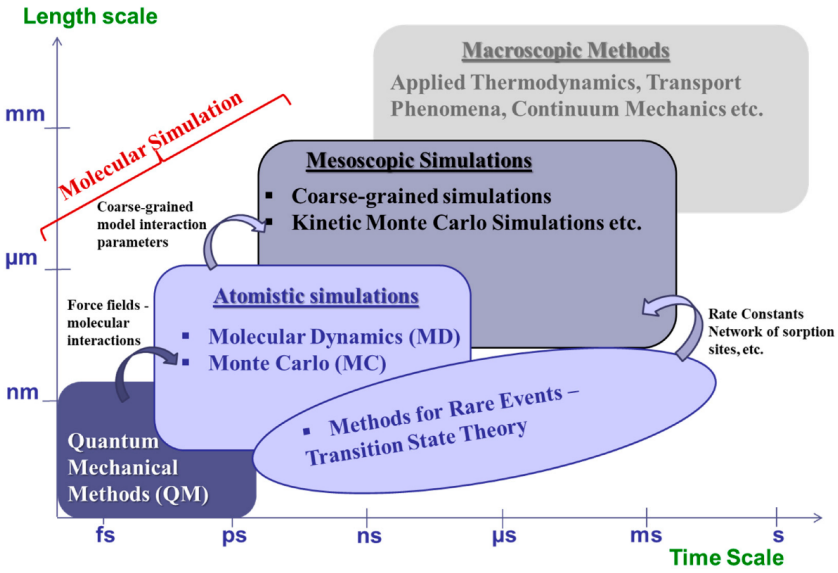


Figure 1: Molecular simulation methods at multiple length and time scales. Hierarchical multiscale simulations utilize information extracted from the previous scale as input for conducting molecular simulations at longer length and time scales [3].

Depending on the problem and phenomena under study and the scales at which these evolve various molecular simulation methods can be applied. Classical simulations utilize classical mechanics and force fields to depict molecular interactions and are employed to study larger systems like proteins and materials over extended periods [1, 22]. In contrast, quantum simulations rely on quantum mechanics to deliver high-precision analyses of smaller systems, such as individual molecules [2]. *Ab initio* simulations for example that belong to the latter category, employ first-principles methods to compute molecular properties. Molecular simulation of complex chemical systems usually necessitates the implementation of hierarchical multiscale methods [23].

MD simulations, which are the most prevalent, involve integrating equations of motion for each atom or molecule in a system. This approach yields data on dynamic molecular behaviour,

including interactions, trajectories, and energies. Other simulation techniques include Monte Carlo (MC) simulations, which use statistical sampling to explore system phase space [1].

Molecular simulations serve as potent tools for understanding intricate molecular systems, offering insights into phenomena like protein folding and material synthesis. They are also valuable for designing and optimizing new materials and drugs, as well as predicting molecular behaviour under various conditions [22]. Nowadays several software packages are available for conducting molecular simulations, such as GROMACS [25], LAMMPS [26], AMBER [27], CHARMM [28], and NAMD [29]. Apart from running MD simulations, some of these packages also include tools for setting up and analyzing and visualizing simulation data [30].

2.1.1 Fundamentals of Molecular Dynamics

MD is one of the widely used molecular simulation methods that enable the study of the time-dependent behaviour of a system of particles by integrating the equations of motion [1, 2]. Appropriate periodic boundary conditions are in many cases applied depending on the problem at hand [22]. During an MD simulation, the positions and velocities of the particles evolve based on a force field that describes the interactions between the particles that constitute the specific system [23]. Macroscopic properties such as pressure, temperature, heat capacity, stress tensor, *etc.*, can be calculated from the microscopic information, according to the fundamental principles of statistical mechanics. MD enable the study of the time evolution of materials and processes provided that the representation in use allows the adequate sampling of phenomena under study based on the relevant length and time scales that take place [31]. MD simulations can be applied at the quantum mechanical (*ab initio* MD, Car-Parinnello MD), atomistic or coarse-grained level and be performed either at equilibrium or out-of-equilibrium conditions [32, 33].

2.1.2 Basic Concepts

Two of the basic elements of the MD simulation are (i) the interaction potential and (ii) the equations of motion governing the dynamics of the particles. Specifically, the 2$^{\text{nd}}$ Newton's law of motion is followed. For a system of N particles Newton's law for each atom i is written as

$$\boldsymbol{F}_i = m_i \boldsymbol{a}_i, \tag{2.1}$$

where m_i is the mass of the particle i, $\mathbf{a}_i$ its acceleration and $\mathbf{F}_i$ is the force acting on it due to the interaction with the other particles of the system. Equivalent the Hamiltonian equation of motion can be solved, where

$$\dot{\boldsymbol{p}}_i = -\frac{\partial H}{\partial \boldsymbol{r}_i}, \tag{2.2}$$

$$\dot{\boldsymbol{r}}_i = -\frac{\partial H}{\partial \boldsymbol{p}_i}, \tag{2.3}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{r}_i$ are the momentum and position coordinated for the i^{th} particle, and H is the Hamiltonian that can be written as

$$H\left(\boldsymbol{p}_i, \boldsymbol{r}_i\right) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + V\left(\boldsymbol{r}_i\right). \tag{2.4}$$

where the $V(\boldsymbol{r}_i)$ is the potential energy in the $\boldsymbol{r}_i$. The force can then be calculated as the derivative of the energy, E, with respect to the change in the particle's position through

$$\boldsymbol{F}_i = m_i \boldsymbol{a}_i = -\nabla_i V = -\frac{dE}{d\boldsymbol{r}_i}. \tag{2.5}$$

By knowing the forces of each particle and the respective masses, the position of each particle at a specific time can then be calculated from the acceleration; the acceleration is the time derivative of the velocity ($\boldsymbol{a}_i = d\boldsymbol{v}_i/dt$) and the position is the time derivative of velocity ($\boldsymbol{v}_i = d\boldsymbol{r}_i/dt$).

Therefore, the outline of the procedure at each time step is that the forces on the atoms are computed and combined with the current positions and velocities to generate the new positions and velocities for the next time step implementing algorithms such as Verlet or velocity Verlet and the new forces are then computed to perform the next cycle [1, 2]. The force acting on each particle is assumed to be constant during the time interval.

There are two important properties of equations of motion that should be noted. The first is that they are time-reversible and the second is that they conserve the Hamiltonian during equilibrium MD simulations [1, 2, 34].

2.1.3 Statistical Ensemble

Statistical mechanics provides a framework for connecting the microscopic details of a system to physical observables such as equilibrium thermodynamic properties, transport coefficients, and spectra [34]. The Gibbs ensemble concept forms the foundation of statistical mechanics, which states that many individual microscopic configurations of a large system lead to the same macroscopic properties [23]. The macroscopic observable properties of a system can be formulated in terms of ensemble averages, where statistical ensembles are usually characterized by fixed values of thermodynamic variables such as the energy, E, the temperature, T, the pressure, P, the volume, V, the particle number, N, or the chemical potential, μ [35].

One fundamental ensemble is called the micro-canonical ensemble and is characterized by constant particle number N, constant volume V, and constant total energy E. This ensemble is denoted as the NVE ensemble [2]. Other examples of statistical ensembles include the canonical or NVT ensemble, the isothermal-isobaric or NPT ensemble, and the grand-canonical or μVT ensemble [36, 37]. These thermodynamic variables that characterize an ensemble can be regarded as experimental control parameters that specify the conditions under which an experiment is performed. This is achieved through the use of thermostats and barostats [38].

Thermostats play a crucial role in maintaining a desired temperature within the system, allowing for simulations in the canonical or NVT ensemble. Different thermostat algorithms, such as Berendsen, Andersen, or Nosé-Hoover methods, are employed to regulate the system's temperature, ensuring that it remains in equilibrium throughout the simulation [39, 40]. Barostats, on the other hand, control the pressure within the system, enabling simulations in the isothermal-isobaric or NPT ensemble. By utilizing barostat algorithms like Berendsen or Parrinello-Rahman methods, the system's volume or cell dimensions can be adjusted to maintain the desired pressure conditions [41, 42]. This is particularly crucial when studying materials subjected to external stresses or investigating phenomena influenced by pressure variations, such as phase transitions or structural changes. The integration of thermostats and barostats into molecular simulations allows the exploration and replication of realistic temperature and pressure conditions, improving the accuracy and relevance of simulation results. By controlling

these thermodynamic variables as experimental parameters, molecular simulations can closely resemble experimental conditions, enabling deeper insights into the behaviour and properties of materials across various scientific and engineering domains.

2.1.4 Inter-Atomic Potentials

The total potential energy is often calculated as the sum of non-bonded and bonded energy contributions. Non-bonded contributions are in many cases extracted using pair potentials that depend only on the distance between atoms to determine van der Waals and electrostatic interactions. Examples of pair potentials are the Lennard-Jones potential, Coulomb potential, Morse potential *etc.* [23].

Bonded contributions may include bond, bond angle, torsion and out-of-plane torsion terms. The total energy of a molecule can then be given by [34]:

$$E = E_{bond} + E_{angle} + E_{torsion} + E_{oop} + E_{cross} + E_{nonbond}, \tag{2.6}$$

where E_{bond} is a pair potential, E_{angle} describes the energy change associated with a change in the bond angle, $E_{torsion}$ describes the energy associated with the rotation between two parts of a molecule relative to each other, E_{oop} describes energy change when one part of a molecule is out of the plane with another, E_{cross} are cross terms between the other interaction terms, and $E_{nonbond}$ describes interaction energies which are not associated with covalent bonding, *i.e.* electrostatic and van der Waal terms [34, 23].

2.1.5 Calculation of Properties

The analysis of the MD simulation results is based on the *ergodic hypothesis* which states that the time average of a physical quantity A equals its ensemble average [34, 23]

$$\langle A \rangle_{time} = \langle A \rangle_{ensemble}\,, \tag{2.7}$$

where the time average can be expressed as

$$\langle A \rangle_{time} = \lim_{\tau \to \infty} \int_{\tau}^{t=0} A\left(\boldsymbol{p}^N(t), \boldsymbol{r}^N(t)\right) dt \approx \frac{1}{M} \sum_{t=1}^{M} A\left(\boldsymbol{p}^N, \boldsymbol{r}^N\right), \tag{2.8}$$

and the ensemble averages

$$\langle A \rangle_{ensemble} = \int\int d\boldsymbol{p}^N d\boldsymbol{r}^N A\left(\boldsymbol{p}^N \mathbf{r}^N\right) \rho\left(\boldsymbol{p}^N \boldsymbol{r}^N\right). \tag{2.9}$$

Based on the above several physical quantities of the system such as its energy, its pressure, its pair correlation function, and its time correlation function can be calculated. The expressions for the calculation of these quantities are followed.

Kinetic Energy In molecular simulations, the kinetic energy of particle i is calculated from its velocity using the formula:

$$E_{kin} = \frac{1}{2} m_i v_i^2, \tag{2.10}$$

where m_i is the mass of the particle i and v_i is its velocity. The total kinetic energy of the system is then obtained by summing over all particles. Once the kinetic energy is known, the

system temperature can be calculated. In addition, the temperature of the system can be monitored during the simulation to ensure that it remains within the desired range and to detect any potential issues, such as energy drift or instability. Overall, the calculation of system temperature is a fundamental aspect of molecular simulations and provides critical insights into the thermodynamics and energy balance of the system.

Average temperature The calculation of average temperature is a critical aspect of molecular simulations, as it provides insights into the thermodynamics and energy balance of the system. In molecular simulations, the system temperature is usually calculated based on the kinetic energy of the particles, which is related to their velocity distribution. The temperature of the system is defined as:

$$T = \frac{2}{3}\frac{E_{kin}}{Nk}, \tag{2.11}$$

where E_{kin} is the total kinetic energy of the system, N is the number of particles, and k is the Boltzmann constant. The factor $\frac{2}{3}$ arises from the equipartition theorem, which states that the average kinetic energy of each degree of freedom in a particle is $\frac{1}{2}kT$. For a system of particles, the average kinetic energy per particle is $\frac{3}{2}kT$, leading to the above formula for temperature.

Pressure tensor

$$\boldsymbol{P}\left(\boldsymbol{r},t\right) = \frac{1}{V}\left[\sum_{i=1}^{N} m_i \boldsymbol{v}_i\left(t\right)\boldsymbol{v}_i\left(t\right)\sum_{j>i}^{N}\boldsymbol{r}_{ij}\left(t\right)\boldsymbol{F}_{ij}\left(t\right)|_{\boldsymbol{r}_i(t)=\boldsymbol{r}}\right], \tag{2.12}$$

where V is the volume, and m_i is the mass of particle i.

Pair Correlation Function The Pair Correlation Function, $g(r)$, is a fundamental property of molecular systems that describes the probability of finding a particle at a certain distance from another particle [1]. It is a key tool in analyzing and understanding the structure and dynamics of molecular systems. In this section, we will discuss how $g(r)$ is calculated from coarse-grained molecular dynamics results.

In coarse-grained molecular dynamics simulations, multiple atoms are grouped together to represent a single particle. This reduces the number of degrees of freedom in the simulation, making it possible to study larger and more complex systems. The $g(r)$ can be calculated from the trajectories generated by these simulations by first calculating the centre-of-mass positions of the particles in the system.

The $g(r)$ is defined as the probability of finding a particle at a distance r from a reference particle, normalized by the number of particles in the system and the volume of the system [1]. Mathematically, the $g(r)$ is given by:

$$g\left(r\right) = \frac{1}{N}\frac{V}{4\pi r^2\Delta r}\Sigma_{i,j}\delta\left(r - r_{ij}\right), \tag{2.13}$$

where N is the number of particles in the system, V is the volume of the system, r_{ij} is the distance between particles i and j, and δ is the Dirac delta function. The term $(\frac{V}{4\pi r^2\Delta r})$ is a normalization factor that accounts for the volume of the shell at distance r, and Δr is the width of the shell.

The $g(r)$ can be calculated using a variety of methods, including histogramming and kernel density estimation [2]. The choice of method depends on the specific application and the quality of the data.

The $g(r)$ provides important information about the local structure and interactions in molecular systems. For example, the peak position and height of the first peak in the $g(r)$ correspond to the average distance between particles and the strength of their interactions, respectively. The shape of the $g(r)$ can also provide information about the presence of ordering or clustering in the system.

Time Correlation Function Time correlation function gives the dynamic and transport properties of the system.

$$K(t) = \lim_{t\to\infty} \frac{\left\langle \left| [A(t) - A(0)]^2 \right| \right\rangle}{2t} = \int_0^\infty d\tau \left\langle \dot{A}(\tau)\, \dot{A}(\tau) \right\rangle, \tag{2.14}$$

where K is the transport coefficient for the physical quantity A, such as the position of the atoms, $r(t)$ in which case the transport coefficient is the diffusion.

Mean Squared Displacement The MSD is a common measure used in molecular simulations to quantify the dynamics and mobility of particles in a system. The MSD is defined as the average squared displacement of particles over a given time interval and is calculated as the difference between the positions of a particle at this interval, squared, and averaged over all particles in the system. The MSD is a useful tool for characterizing the diffusional motion of particles in a system, and it can provide insights into the nature of the interactions between particles and the underlying energy landscape. In addition, the MSD can be used to estimate the diffusion coefficient based on the Einstein relation provided that a Fickian regime is reached that is identified from a slope equal to unity in log(MSD)-log(t) plot. Overall, the mean square displacement is a versatile and powerful tool for analyzing the dynamics and transport properties of particles in molecular systems, and it has many applications in both theoretical and experimental studies.

The MSD is calculated by measuring the displacement of each particle over a given time interval, squaring the displacements, and then averaging the results over all particles in the system. The formula for the MSD is:

$$MSD(t) = < |r(t+\tau) - r(t)|^2 >, \tag{2.15}$$

where $r(t)$ is the position vector of a particle at time t, and $< |...| >$ denotes the ensemble average over all particles in the system. The time interval τ represents the lag time or time difference between the two positions, and $t+\tau$ is the later time at which the position is measured. The MSD provides a measure of the average squared displacement of a particle over time, which can be related to the diffusion coefficient and other transport properties of the system. The MSD can be calculated for a range of lag times and plotted as a function of time to obtain insights into the dynamics and mobility of particles in the system. Overall, the calculation of the mean square displacement is a fundamental tool for studying the dynamics and transport properties of molecular systems, and it has many applications in both theoretical and experimental studies.

2.1.6 Coarse-Grained Molecular Simulations

In CG modelling generally, the atomistic systems are represented with fewer degrees of freedom [43], enabling the study of phenomena at longer lengths and time scales. CG methods are in many cases the core of multiscale modelling methods that are necessary for the molecular

simulation of complex chemical systems such as macromolecular ones. Coarse-Grained Molecular Dynamics (CG-MD) for example has been used to tackle the high computation cost that the all-atom MD has CG methods may include methods that describe the reduced representation in molecular structures, schemes assuming atomic-scale homogeneous structures, and methods that use field representation.

CG methods could be divided in the following categories:[44]:

1. those that describe the reduced representation in molecular structures
2. those assuming atomic-scale homogeneous structures
3. those using field representation.

In the field of soft matter, most CG models are of the first type. This type contains two processes. CG methods, in this case, involve the determination of (i) an appropriate mapping between the higher resolution and the lower resolution level based on the chemical system and the problem at hand, (ii) the force field that is able to describe the interactions between the moieties at the coarse-grained level and (iii) in many cases, the reverse mapping back to the more detailed representation.

In general, there is no unique way to coarse-grain. The coarse-graining strategies should reduce the number of system degrees of freedom [45], by substituting groups of atoms with single interaction sites, while maintaining the ones that are important for the description of the mechanisms/processes under study [3]. Therefore, the CG model used in each case is related to the molecular system and the scientific problem under study, making the mapping to the CG representation largely empirical. The main approach for the development of CG force fields can fall into two categories namely top-down [46] or bottom-up approaches [47]. In the former, the reproduction of the macroscopic properties of the molecular system is targeted while in the latter the reproduction of microscopic properties of the molecular system, based on atomistic simulations or experimental findings, is aimed.

2.2 Machine learning

ML is a subdomain of Artificial Intelligence (AI) that focuses on developing algorithms and models capable of learning from data to make predictions or decisions. The growth of ML has been substantial in recent years, fueled by the increasing availability of data and computational resources.

Three primary types of machine learning exist, supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training algorithms on labelled datasets with known outputs, enabling them to make predictions on new, unlabeled data [48, 49]. In unsupervised learning, algorithms are trained on unlabeled datasets, to discover patterns or structures in the data. Reinforcement learning entails algorithms learning to make decisions based on feedback from their environment.

ML has been effectively applied across various fields, including image and speech recognition, natural language processing, robotics, and materials design and discovery. One characteristic example in the latter domain with significant progress in applications related to pharmaceutical research, where ML has been utilized to predict drug molecule properties, identify new drug targets, and design compounds with specific characteristics [50, 51, 52].

Common ML algorithms encompass Decision Trees, Support Vector Machines (SVM), Random Forests, and Neural Networks. Deep Learning models, such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), are also widely employed. These algorithms cater to a wide range of tasks, including classification, regression, and clustering.

Despite ML's successes, challenges in its implementation remain. These include selecting suitable algorithms and models for a specific task, collecting and preparing high-quality data, and avoiding overfitting or underfitting the model. To address these challenges, researchers have developed various techniques, such as cross-validation, regularization, and ensemble methods [53, 54, 55]. One of the major open challenges in applying ML in natural sciences and technology is to appropriately develop schemes that efficiently incorporate physical laws and domain knowledge, a fact that is crucial due to the inherent agnostic character of the ML techniques.

ML is currently investigated to be applied in numerous scientific domains, including materials science and engineering. ML has been employed in the materials informatics domain to extract various properties, such as molecular structure, binding affinity, and solubility, or to model reaction kinetics obtaining interesting results [50, 51]. Owing to its ability to capture complex, nonlinear relationships between molecular features and properties, ML has the potential to complement and advance "traditional" computational methods based solely on physical principles [53].

One application of ML in molecular simulation materials design involves predicting molecular properties. ML methods can construct its incorporation in Quantitative Structure-Activity Relationship (QSAR) models that predict the properties of new molecules based on their structural features. These models can screen extensive compound libraries for potential drug candidates or optimize the properties of existing compounds [51]. For example, DeepChem has been used to predict the activity of small molecules against various protein targets [56].

There are several rapidly growing intersections of machine learning, materials informatics and molecular simulations which include the utilization of molecular simulation data to develop materials informatics schemes, the development of ML methods for the interpretation and post-processing of molecular simulation results, the implementation of ML methods for analytical force fields parametrizations as well as for the study of rare events and rate-limited processes such as in protein folding [55].

One of the potential applications of ML that is currently explored is its integration molecular simulation via its utilization in the development of force fields. Traditional classical force fields, utilize predefined functional forms to describe interactions between particles (*e.g.* atoms or groups of atoms). However, these functional forms lack the flexibility required to accurately capture intricate interactions and capture complex energy hypersurfaces and many-body effects. ML methods can be used to learn interactions directly from data [52]. For instance, SchNet has been utilized to develop a deep neural network-based force field for organic molecules, accurately predicting molecular energies and forces [56]. Despite some interesting results that have appeared in the literate in the recent few years, there are still several challenges and pitfalls that have to be addressed in order to be able to develop and use ML-based force fields for the conduction of meaningful (atomistic or coarse-grained) molecular simulations of bulk material systems [18, 57]. This remains a very challenging task even for simple chemical systems in the bulk.

Additionally, despite ML's potential in this field, several general obstacles such as developing more interpretable ML models, choosing appropriate descriptors, preserving physical laws and symmetries and constructing adequate loss functions that will enable the identification of a successful training process which is crucial and is currently missing.

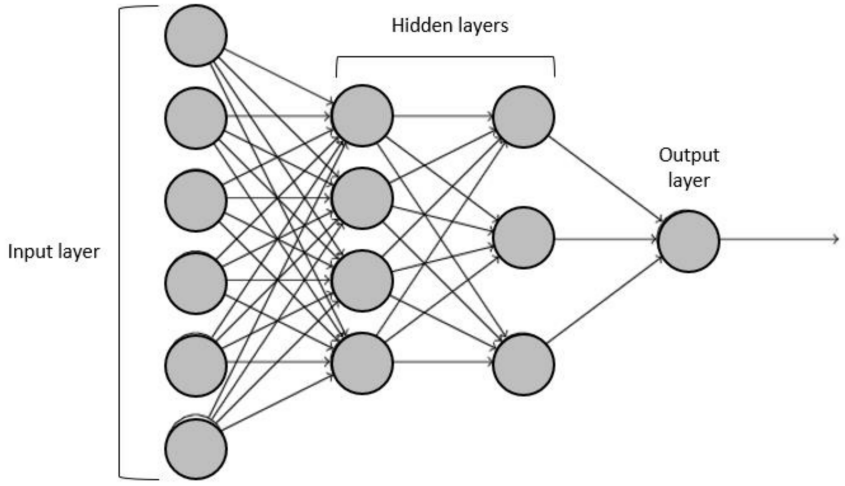


Figure 2: Geometry of a multi-layer artificial neural network (ANN) [59].

One application of ML in molecular simulations is the development of force fields. Traditional force fields, such as CHARMM and AMBER, utilize simple functional forms to describe interactions between atoms or groups of atoms. However, these functional forms lack the flexibility required to accurately capture intricate interactions like non-covalent interactions or polarization effects. ML methods can learn force fields directly from data, enabling more precise and efficient simulations [52]. For instance, SchNet has been utilized to develop a deep neural network-based force field for organic molecules, accurately predicting molecular energies and forces [56].

Another application of ML in molecular simulations involves predicting molecular properties. ML methods can construct QSAR models that predict the properties of new molecules based on their structural features. These models can screen extensive compound libraries for potential drug candidates or optimize the properties of existing compounds [51]. For example, DeepChem has been used to predict the activity of small molecules against various protein targets [56].

In addition to force field development and property prediction, ML has been applied to other challenges in molecular simulations, such as structure determination, molecular dynamics simulations, and protein folding [55]. However, despite ML's potential in this field, several obstacles persist. These include developing more interpretable ML models, integrating ML with existing simulation methods, and generating high-quality training data [54].

2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a machine learning method, in which specific tasks are learned to be performed by analyzing training examples. The neural networks, inspired by the brain function principle [1], which has a high number of processing nodes that are densely interconnected. The number of processing nodes can range from one up to several million with the upper limit set by the level of abundance of processing resources.

A schematic representation of a simplified ANN geometry is presented in Figure 2. The ANN can be represented as a network of nodes with direct links. Each node corresponds to an

[1] This is a very loose comparison for the easy illustration of the Artificial Neural Network (ANN). The literature supports that the mammal's brain is so advanced that its processing power cannot be compared even with that of the most advanced computer [58].

artificial neuron. The direction of the information in the ANN is indicated by the edge of the links.

As the Perceptron is the simplest ANN (Figure 3) can be a simplified introduction to the working principles of the ANN nodes.

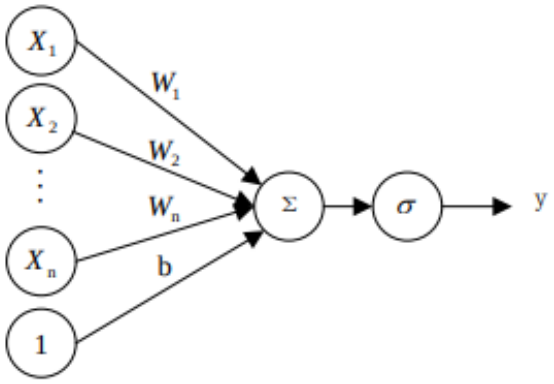


Figure 3: Perceptron model.

The Perceptron has two layers, the input layer and the output layer, that are directly connected to each other and can be described by:

$$y = \sigma(\mathbf{W}^T\mathbf{X}), \tag{2.16}$$

where **X** represents vector $[X_1, X_2, ..., 1]$, **W** represents vector $[W_1, W_2, ..., b]$, and σ represents activation function. Thus the output of the Perceptron is the result of the activation function when its input is the input vector multiplied with some weights. The target of the training of the Perceptron is to find the optimum values of these weights to describe the training examples. The training algorithm for the Perceptron follows the steps below.

1. Initialise **W**
2. while True
   (a) Error = 0
   (b) for (x, y) in $(x(i), y(i)(i = 1, .., N)$
      i. if $\mathbf{W}^T\mathbf{X} < 0$:
         A. $\mathbf{W} += yx$
         B. Error +=1
   (c) if Error == 0
      i. break

The limitation of the structure of Perceptron sets boundaries in its practical applications. The original Perceptron can only deal with linear separable problems, whereas for linearly non-separable problems, the algorithm will oscillate in the calculation process. Although, the Perceptron can also be considered as the building block of ANN.

2.2.2 Convolutional Neural Networks (CNNs)

CNNs have recently revolutionized computer vision tasks, including image analysis and recognition, and are increasingly being applied in other domains such as natural language processing [60]. The architecture of a typical CNN consists of multiple layers, each designed to perform a specific function [53]. There are three main types of layers in a CNN: convolutional, pooling, and fully connected [61].

Convolutional layers are the key building blocks of CNNs. They apply a set of learnable filters to the input image, generating feature maps that capture different aspects of the image. During the convolution process, the filters slide over the input image and compute the dot product of the filter weights and the corresponding pixel values. The resulting feature map represents the activations of the filter at each location in the input image. Convolutional layers can be followed by activation functions such as Rectified Linear Unit (ReLU), which introduce nonlinearity into the network, and normalization layers such as batch normalization, which improve the stability and performance of the network.

Pooling layers reduce the spatial dimensions of the feature maps produced by the convolutional layers while retaining the most critical information. The most common type of pooling is max pooling, which partitions each feature map into non-overlapping rectangular regions and computes the maximum value within each region. Other types of pooling include average pooling and L2 pooling.

Fully connected layers are similar to those in traditional neural networks and are typically used at the end of the CNN to produce the final output. They connect every neuron in one layer to every neuron in the next layer, and their weights are learned during training.

CNNs are trained using backpropagation, which involves computing the gradient of a loss function concerning the weights of the network and updating the weights using an optimization algorithm such as Stochastic Gradient Descent (SGD). The loss function is typically a measure of the difference between the predicted output of the network and the true output and is minimized during training.

CNNs have been applied to a wide range of applications, including image classification [62], object detection [63], segmentation [64], and synthesis [65], as well as natural languages processing tasks such as sentiment analysis [66] and machine translation [67]. Their ability to automatically learn features from raw data has significantly improved performance on a wide range of computer vision tasks. As the field of deep learning continues to evolve, it is likely that CNNs will continue to play an important role in many applications [53].

2.2.3 Graph Neural Networks

Graph Neural Network (GNNs) are a type of neural network designed to process graph-structured data [68]. A graph is a set of vertices (or nodes) connected by edges and is a commonly used data structure in a variety of fields, including social networks [69], biology [70], chemistry [71], and recommendation systems [72].

GNNs aim to learn useful representations of graph-structured data by aggregating information from the local neighbourhoods of each node [68]. The basic idea is to propagate information through the edges of the graph, using a series of neural network layers. At each layer, a node aggregates information from its neighbours, typically by taking a weighted sum of their features. This information is then passed through a nonlinear activation function and used to update the node's own feature representation.

Several variations of GNNs have been proposed, including Graph Convolutional Networks (GCNs) [73], Graph Attention Networks (GATs) [74], and GraphSAGE [75]. GCNs use a

convolutional operation to aggregate information from neighbouring nodes, while GATs use attention mechanisms to weigh the importance of each neighbour. GraphSAGE is a general framework that allows for flexible neighbourhood aggregation strategies.

GNNs have been shown to be effective for a wide range of tasks, including node classification, link prediction, and graph classification. They have also been applied to real-world [76], and social network analysis [77].

2.2.4 Graph Convolutional Neural Network

GCNN are an efficient variance of CNN on graphs. GCNN stack layers of learnable first-order spectral filters followed by nonlinear activation function to learn graphs representations [78].

The GCNN falls into two categories, the spectral-based and the spatial-based [79]. In the first one, the graph convolutions are defined by introducing filters from the perspective of graph signal processing, and the graph convolution operation is interpreted as removing noise from the graph signal [80]. In the second one, graph convolutions are defined by information propagation [79].

2.3 Multicomponents loss functions

In machine learning model training, optimizing an objective function that is defined as a weighted linear combination of multiple losses is a common problem [81]. The performance of the final model can be highly dependent on the choice of weights assigned to each loss. Traditionally, the optimal set of weights is determined by setting them as hyperparameters[2] and conduct an exhaustive grid search[3], which can be computationally expensive. However, in recent years, several weight adaptation methods have been proposed that can be applied during network training. In the following section, we will provide a description of Multi-Objective Optimization (MOO) and a brief overview of the most widely used weight adaptation methods in physical problems.

2.3.1 Multi-Objective Optimization (MOO)

MOO targets simultaneous optimisation of a set of $k > 1$, that can potentially be conflicting objectives [83]:

$$\mathcal{L}(\theta) = \left(\mathcal{L}_1(\theta), ..., \mathcal{L}_k(\theta)\right)^T, \tag{2.17}$$

which can be turned into a single objective through linear scalarization:

$$\mathcal{L}(\theta) = \sum_{i=1}^{k} \lambda_i \mathcal{L}_i(\theta), \lambda_i \in R_{>0}. \tag{2.18}$$

Many problems in engineering, natural sciences, or economics can be formulated as multi-objective optimizations and generally requires trade-offs to satisfy all objectives to a certain degree simultaneously. The solution of MOO models is usually expressed as a set of Pareto

[2] In machine learning, hyperparameters are parameters that are set before the learning process begins and remain fixed throughout the training process [82]. They determine the behaviour and performance of the learning algorithm rather than being learned from the data itself

[3] Grid search is a technique used in hyperparameter optimization to systematically explore a predefined set of hyperparameter combinations for a machine learning model. It is a brute-force method that exhaustively searches the entire hyperparameter space by evaluating the model's performance for each combination.

optima, representing these optimal trade-offs between given criteria according to the following definition [84].

Definition 2.1. A solution $\hat{\theta} \in \Omega$ Pareto dominates solution θ (denoted $\hat{\theta} \prec \theta$ if and only if $\mathcal{L}_i\left(\hat{\theta}\right) \leq \mathcal{L}_i\left(\theta\right), \forall i \in 1, ..., m$ and $\exists j \in 1, ..., m$ such that $\mathcal{L}_j\left(\hat{\theta}\right) < \mathcal{L}_{ij}\left(\theta\right)$.

Definition 2.2. A solution $\hat{\theta} \in \Omega$ is said to be Pareto optimal if $\forall \theta \in \Omega, \hat{\theta} \preceq \theta$. The set of all Pareto optimal points is called the Pareto set and the image of the Pareto set in the loss space is called the Pareto front.

Theoretically, a Pareto optimal solution θ is independent of the scalarization [85]. However, when using neural networks for MOO, the solution space becomes highly non-convex. Thus, although neural networks are universal function approximators [86], they are not guaranteed to find the globally optimal solution through gradient-based optimization. Scaling the loss space, therefore, provides the option of guiding the gradients into having an a priori deemed desirable property. However, manually finding optimal λ_i requires a laborious grid search and becomes intractable as k gets large. Furthermore, one might want to let λ_i evolve over time. This raises the need for an automated scheme to dynamically choose the scalings of λ_i.

2.3.2 Self-balancing methods

Three widespread loss self-balancing methods are the Learning Rate Annealing [87] from the field of Physics-Informed Neural Networks (PINN) [88] and the GradNorm [15] and SoftAdapt [16] originating from Computer Vision.

Learning Rate Annealing The Learning Rate Annealing method, proposed by *Wang et al.* [87], addresses the imbalance of the different loss terms in PINN used for finding the unknown, underlying function able to solve parameterised Partial Differential Equation (PDE), through adaptively scaling the loss using gradient statistics. In such cases, the loss components are the $\mathcal{L}_\Omega$, the $\mathcal{L}_{\Gamma_i}$, the $\mathcal{L}_{\Upsilon_i}$, and the $\mathcal{L}_\Psi$ and maps to the governing equation, the boundary conditions, the initial conditions, and the data (Figure 4).

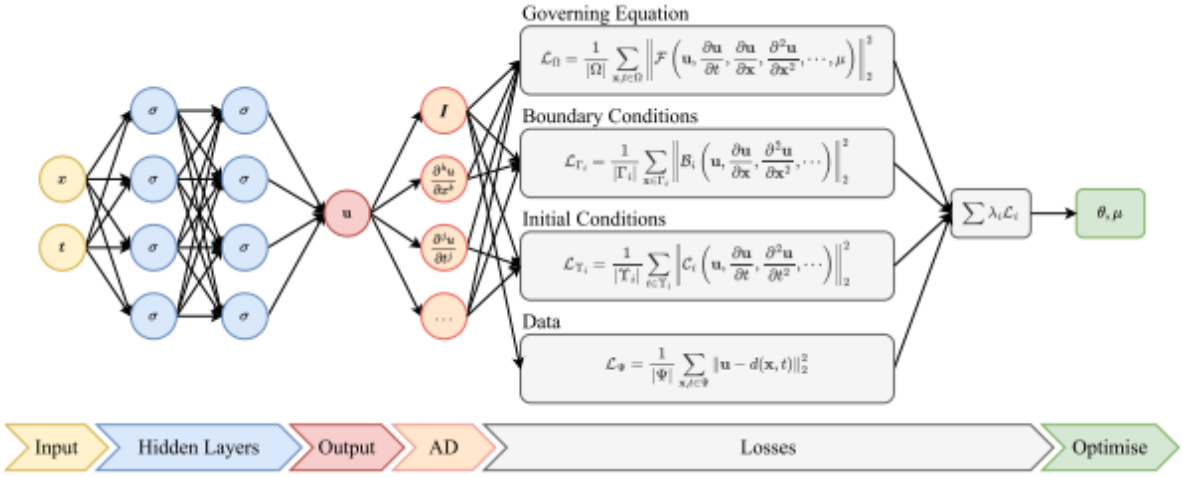


Figure 4: Schematic of a PINN: A fully-connected feed-forward neural network with space and time coordinates (x, t) as inputs, approximating a solution $\hat{u}\left(x, t\right)$. [82].

Following the above description the implementation of the Learning Rate Annealing leads to the calculation of the coefficient of the i^{th} loss component at the t epoch through:

$$\begin{aligned} \hat{\lambda}_i(t) &= \frac{max\{|\nabla_\theta \mathcal{L}_\Omega(t)|\}}{\overline{|\nabla_\theta \mathcal{L}_{\{\Gamma,\Upsilon\}_i}(t)|}}, i \in \{1,...,k\} \\ \lambda_i(t) &= \alpha\lambda_i(t-1) + (1-\alpha)\hat{\lambda}_i(t). \end{aligned} \tag{2.19}$$

where $\overline{\left|\nabla_\theta \mathcal{L}_{\{\Gamma,\Upsilon\}_i}(t)\right|}$ is the mean of the gradient w.r.t the parameter θ and α is a hyperparameter.

With this method, whenever the maximum value of $\left|\nabla_\theta \mathcal{L}_{\{\Gamma\}_i}(t)\right|$ grows considerably larger than the average value in $\overline{\left|\nabla_\theta \mathcal{L}_{\{\Gamma,\Upsilon\}}(t)\right|}$, the scalings $\hat{\lambda}_i(t)$ correct for this discrepancy such that all gradients have similar magnitudes. Additionally, exponential decay is used in order to smoothen the balancing and avoid drastic changes in the loss space between optimisation steps.

GrandNorm In the case of GrandNorm method the loss coefficients are trainable [15] . The update of the loss coefficients is performed by a separate optimiser and is such that all the terms improve at the same relative rate with respect to their initial loss [15]. The loss function of this optimizer is expressed as

$$\mathcal{L}(\lambda;t) = \sum_{i=1}^{k} \left| G_\theta^{(i)}(t) - \overline{G}_\theta(t) \times [r_i(t)]^\alpha \right|_1 \tag{2.20}$$

where $G_\theta^{(i)} = G_\theta^{(i)} \left\|\nabla_\theta \lambda_i \mathcal{L}_i(t)\right\|_2$ is the L_2 norm of the gradient w.r.t. the network parameter θ for the i^{th} loss coefficient, $\overline{G}_\theta(t) = \frac{1}{k}\sum_{i=1}^{k} G_\theta^{(i)}$ is the average of all gradient norms, $r_i(t) = \mathcal{L}_i(t)/(\mathcal{L}_i(0)\cdot\overline{\mathcal{L}}(t))$ defines the rate at which term i improved so far, and α is a hyperparameter representing the strength of the restoring force which pulls tasks back to a common training rate. The final loss for the update of the network parameter is then simply linear scalarization using the updated coefficients:

$$\mathcal{L}(\lambda;t) = \sum_{i=1}^{k} \lambda_i \mathcal{L}_i(t). \tag{2.21}$$

SoftAdapt In SoftAdapt method as in GrandNorm the similar relative rate of progress of the different terms is targeted [16]. Although, in this case, only the previous time step is taken into account. The loss coefficients are calculated as

$$\lambda_i(t) = \frac{\exp\left(\mathcal{T}\left(\mathcal{L}_i(t) - \mathcal{L}_i(t-1)\right)\right)}{\sum_{j=1}^{k}\exp\left(\mathcal{T}\left(\mathcal{L}_j(t) - \mathcal{L}_j(t-1)\right)\right)}. \tag{2.22}$$

2.4 Related work

The study of existing methodologies for the balancing of loss coefficients in deep learning models—particularly in relation to molecular dynamics and physical issues—inevitably warrants the creation of a robust framework for comparison. Manual tuning, grid search, random search, Bayesian optimization, and multi-task learning are among the techniques currently employed [89, 90, 91, 84]. Each approach has its own unique benefits and potential drawbacks.

For example, manual tuning, albeit simple, can be a laborious and potentially suboptimal process due to its reliance on human intuition and trial-and-error. Automated techniques like

Bayesian optimization and multi-task learning offer streamlined processes, yet they might necessitate greater computational resources and meticulous execution to yield optimal results.

Despite the availability of these methods, there remains a critical need to establish a more robust, quantitative, and reproducible methodology for comparing various loss coefficient balancing techniques—one that is application-agnostic. This necessity stems from the requirement to extrapolate comparisons across a wide spectrum of applications, thus ensuring the replicability of outcomes. Crucially, a robust comparative methodology should offer transparent statistical figures of merit, allowing the performance of distinct approaches to be evaluated based on tangible physical parameters.

Existing methods tend to rely on model-specific metrics such as accuracy or loss. However, these metrics may not always correspond to physically meaningful outcomes. Thus, the adoption of a method that employs physical parameters as key figures of merit allows for a more direct appraisal of a model's applicability to physical problems. For instance, in the realm of molecular dynamics simulations, the comparison could hinge on figures of merit such as the predicted potential energy surface, forces, or the long-term stability of the system being simulated [92].

Therefore, the formulation of a robust, quantitative, and reproducible comparative methodology that uses physical parameters as figures of merit stands to significantly contribute to the rigorous assessment and selection of the most suitable loss coefficient balancing approach for deep learning models in molecular dynamics and physical problems.

The primary objective of this research endeavour is to formulate a robust and repeatable methodology, capable of automating the pre-screening of insufficiently trained models, while also quantifying the performance of more promising models. This is pursued to establish objective and reliable benchmarks for assessing various models, particularly within the realms of molecular dynamics and physical problems, where accurate and precise model predictions are paramount.

To illustrate this concept, we have employed self-balancing techniques for the components of a multi-objective loss function during the training of a GCNN model. The model was designed to describe coarse-grained interactions - an area of molecular dynamics where striking a balance among various loss components is crucial for generating accurate and meaningful predictions. The loss function in this instance is composed of two components. The first component targets the training on the obtained coarse-grained forces of coarse-grained benzene particles in a single bead representation. The second component focuses on the intermolecular potential energy of the system, with respect to the ground truth values ascertained from atomistic simulations of the bulk benzene liquid system [93].

These examples serve as tangible explorations into developing an automated, rigorous method for model pre-screening and performance quantification. The insights gained from these endeavours, particularly with respect to the balancing of loss coefficients, are pivotal in refining our proposed methodology. The ultimate goal is to create a universally applicable approach for evaluating deep learning models used in molecular dynamics and physical problems. Further details on the model and the training datasets used in this research are provided in Sections 3.1 and **??**, respectively.

3 Methodology

This research addresses the problem of balancing loss coefficients in multi-component functions of ML architectures. Imbalanced loss terms during model training can lead to gradient pathologies and performance failure [82]. Although several self-balancing approaches have been proposed (Section 3.3), a robust and efficient method for determining the optimal approach is lacking. This research aims to develop a methodology to identify the most suitable self-balancing approach, improving model performance and providing insights for handling loss coefficient imbalances in ML architectures. By addressing this problem, the thesis contributes to enhancing the training process and overall accuracy of ML models.

3.1 Machine learning model

The machine learning model used in this work is based on the SchNet, [94, 95] and modified to develop CG Machine Learned potentials for bulk organic systems using liquid benzene as a test system and implementing a strategy that includes a force-matching scheme [93]. A schematic illustration of the SchNet model is presented in Figure 5.

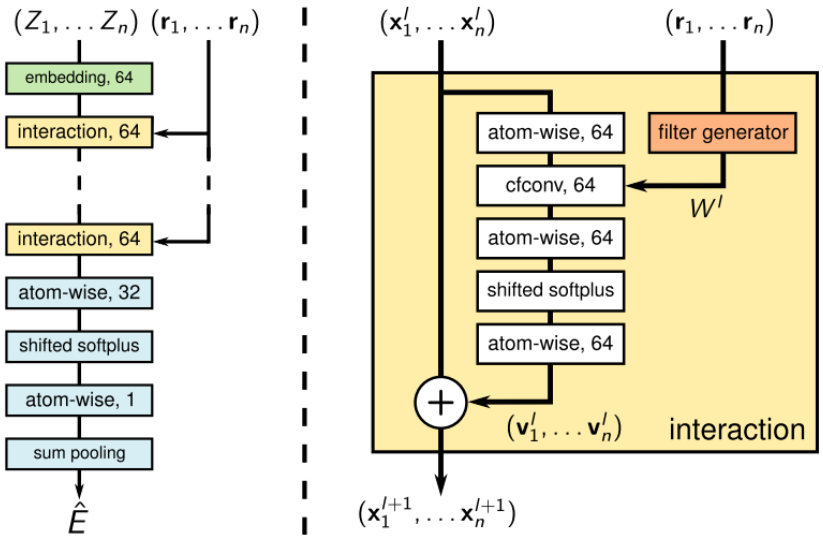


Figure 5: Illustrations of the SchNet architecture (left) and interaction blocks (right) with atom embedding in green, interaction blocks in yellow, and property prediction network in blue. For each parameterized layer, the number of neurons is given [95].

In this model, each particle is represented through a vector, which is initialized to distinguish between the particle's chemical identities (embedding layer). The feature is then updated in each neural network layer depending on the chemical environment, by performing continuous convolutions across the particle neighbourhood and optimizing the convolutional filter weights during the training (convolutional layers). Afterwards, a fully connected section is included (readout layers). A sequence of continuous convolution layers and readout layers constitutes a SchNet "block". Multiple blocks can be utilized in series to define the full network architecture.

The output found at the end of the blocks can be interpreted as a learned feature representation, which encodes the many-body information from the particle neighbourhood required to predict the target property. Finally, a fully connected (dense) section transforms the fingerprints into the final scalar output, which is interpreted as a per-particle energy contribution. This local decomposition ensures invariance of the ML potential architecture to the total number of

atoms. All energy contributions are summed to obtain the total energy ($\sum_i \hat{U}_i$), which is then differentiated with respect to the positions of the particles to predict the force acting on each particle. These are the target properties for the network training.

For the single bead CG representation for each benzene molecule, the network optimization is performed by minimizing a loss function consisting of two terms. The one term corresponds to the mean squared difference between the predicted forces and the forces associated with the MD trajectory considered as input and the second one is related to the intermolecular energy of the system. The second term was necessary to be included in the loss as no meaningful ML-based models could be extracted for a wide hyperparameters search in the case that only forces were taken into account during training for this system in which only intermolecular interactions were present at the CG level [17]. The loss function is formulated as

$$\mathcal{L} = \mathcal{L}_F + \mathcal{L}_U = \left(\nabla_{r_i} \left(\sum_i \hat{U}_i + U_{ex} \right) - F\left(r_i\right) \right)^2 + \lambda \left(\left(\sum_i \hat{U}_i + U_{ex} \right) - U_{int} \right)^2, \quad (3.1)$$

In the loss function a regularization term, related to non-bonded repulsion terms, imposes excluded volume effects. The purpose of including such a constraint term is to ensure that the energy of the system will be driven to infinity if nonphysical states occur that are not within the training data. In particular, an excluded volume energy term based on the pairwise distances between the moieties is considered:

$$U_{ex} = \sum_{i=1}^{N} \sum_{j=i}^{N} \left(\frac{\sigma}{\|x_i - x_j\|} \right)^{n^{ex}}, \quad (3.2)$$

where σ and n^{ex} are hyperparameters of the model, for which suitable values must be identified. This term is added to the total energy predicted by the GCNN, prior to differentiation.

3.2 Experimental Setup

Dataset Atomistic MD simulations had already been performed on three independent fully atomistic liquid benzene systems containing 500 molecules each, at 300 K and 340 K (Figure 6) that have served as training data for the models. The systems were initially equilibrated through a 1 ns NPT simulation, and then a 4 ns NVT run at the average equilibrium density was conducted. The first ns of the run were discarded, and, after that, 20000 configurations, saved every 1 ps, were retained for each system, and constituted the training data for the force field development. The first 9000 of them were used in the training of the model. Each benzene molecule was mapped into a single CG site. This choice allows the study of the application of the ML method to a CG system that contains only intermolecular interactions, which are the most complex to represent and the ones for which the expressive power of a Neural Network (NN) model could provide a greater advantage compared to traditional CG models. NVT CG simulations with the ML potentials had already been conducted at 300 K using the ASE integrator [96] to validate the extracted ML-based and to optimize accordingly the hyperparameter set to used [17]. Details on the methodology and full set of hyperparameters can be found in Ref [17].

Model Evaluation For the assessment of the performance of each model, the potential energy, mean square displacement, pair correlation function, and temperature calculated from

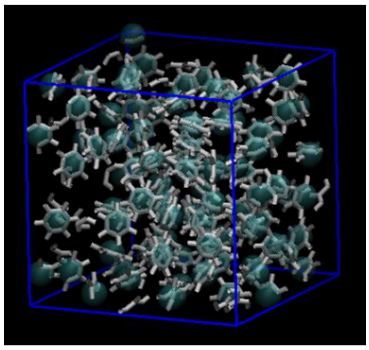


Figure 6: System studied: 500 molecules of liquid benzene at 300 K mapped onto one CG bead each (light blue spheres) [17].

simulation results were monitored and compared against the atomistic simulation results. Details about these physical parameters and method of calculation are presented in Section 2.1.5. Statistical parameters of these results are extracted for comparison and evaluation purposes.

The statistical parameters that are used are:

- the absolute percentage difference between the mean value of the coarse-grained and the atomistic simulated potential energy of the system.
- The coefficient of variation of the coarse-grained simulated potential energy of the system
- The mean coarse-grained simulated temperature of the system
- The coefficient of variation of the coarse-grained simulated temperature of the system
- The statistic and p-value of Kolmogorov–Smirnov test between the coarse-grained and atomistic simulated pair correlation functions.
- The slope of the last 30 % of the mean square displacement of the curve.

The above parameters were selected such as the performance of the model can be quantified and the conclusion on its succession derived without the need of exhausting visualization of the results. Specifically, the second and fourth parameters indicate the stability of the simulation results through the calculation of the Coefficient of Variation (CV) of CG potential energy and temperature, the absence of which is related to model failure. In the same spirit, the last parameter assesses the state of a system with a slope reaching 0 indicating an unstable simulation that resulted in unphysical states of systems with no CG particle mobility. If these first levels are determined as successful, then both the first and third parameters are measured to indicate the validity of the results as the values of the mean potential energy and temperature from coarse-grain simulations via comparison with the atomistic simulations results. The fifth parameter indicates the similarity between the coarse-grained and atomistic simulated pair correlation functions. This is evaluated employing Kolmogorov-Smirnov (K-S) test. Additional parameters indicating the efficiency of the training process are also used for the model evaluation. These parameters are the number of epochs in the training and the time of each training epoch.

In the above two statistical analysis methods are employed, namely coefficient of variation and Kolmogorov-Smirnov test. We elaborate on these two methods in the following paragraphs.

Coefficient of variation The coefficient of variation is a useful statistical measure that quantifies the relative variability of a dataset. It is particularly valuable when comparing the dispersion of datasets with different units of measurement or differing magnitudes. It is defined as the ratio of the standard deviation (σ) to the mean (μ) of a dataset, usually expressed as a percentage (eq. 3.3).

$$CV = 100 \cdot \frac{\sigma}{\mu}. \tag{3.3}$$

The coefficient of variation provides insights into the relative variability of a dataset. A higher CV value indicates a larger degree of dispersion or variability relative to the mean, while a lower CV value signifies a smaller degree of dispersion.

Kolmogorov-Smirnov test The Kolmogorov-Smirnov test is a non parametric statistical method for assessing the goodness-of-fit between two probability distributions or testing the null hypothesis that a sample is drawn from a specific distribution [97]. It is based on the comparison of the Empirical Cumulative Distribution Function (ECDF) of a sample with either another ECDF from a different sample (two-sample K-S test) or a theoretical Cumulative Distribution Function (CDF) from a reference distribution (one-sample K-S test). The test statistic is the maximum absolute difference between the two compared CDFs. The K-S test yields a test statistic (D) and a p-value. The null hypothesis states that the sample data are drawn from the reference distribution (one-sample K-S test) or that the two samples come from the same underlying distribution (two-sample K-S test). If the p-value is smaller than a predetermined significance level (*e.g.*, 0.05), the null hypothesis is rejected, indicating that the sample data do not follow the reference distribution or that the two samples have different underlying distributions.

3.3 Loss components value and gradient based approaches

In the framework of the present thesis, different self-balancing approaches based on the values loss components and three variations of a self-balancing method based on the gradient of the loss components, namely SoftAdapt, were employed as alternatives to the constant loss components balancing. In the three value-based approaches, the loss term coefficients are calculated by the ratio of their values in the previous epochs, their overall maximum values and their maximum values of the latest N epochs are used.

Mathematically, some indicative predefined approaches for a loss function consisting of k terms can be expressed as:

$$\begin{aligned} \lambda_i(t) &= \tfrac{\mathcal{L}_i(t-1)}{\mathcal{L}_k(t-1)}, i = 1, .., k-1 && \text{ratio,} \\ \lambda_i(t) &= \tfrac{1}{max([\mathcal{L}_i(1), ..., \mathcal{L}_i(t-1)])}, i = 1, .., k && \text{overall max,} \\ \lambda_i(t) &= \tfrac{1}{max([\mathcal{L}_i(t-N), ..., \mathcal{L}_i(t-1)])}, i = 1, .., k && \text{N previous,} \end{aligned} \tag{3.4}$$

where the $\lambda_i(t)$ is the coefficient of the i^{th} loss term coefficient that is used in the t^{th} epoch. In the case of the ratio approach, the coefficients of all the loss terms are normalized regarding a selected one, whereas in the other two cases, no information regarding the value of the loss term is shared for the calculation of the coefficients.

The SoftAdapt family of methods that were also employed in this research was developed by *A. Ali Heydari et al.* [16] to address the problem of the weighting of the multi-part loss functions and implemented in image reconstruction and synthetic data generation. The methods, inspired by Softmax, adaptively update the weights of the linear combination of individual objective

functions, depending on the performance of each part and the collective loss function as a whole. SoftAdapt evaluates the performance by approximating the rate of change of each loss function over a short history, which indicates if it has been increasing or decreasing. SoftAdapt then compares the individual rates of change and determines how visible each objective function should be to the optimizer. The SoftAdapt family consists of three methods, the original variant, the weighted, and the normalized.

Mathematically, the three methods are described as:

$$\begin{aligned}
\lambda_i\left(t\right) &= \frac{\exp(\beta s_i(t)))}{\sum_{j=1}^{k}\exp(\beta s_j(t)))}, i=1,..,k && \text{Original Variant,}\\
\lambda_i\left(t\right) &= \frac{\mathcal{L}_i(t)\exp(\beta s_i(t)))}{\sum_{j=1}^{k}\mathcal{L}_j(t)\exp(\beta s_j(t)))}, i=1,..,k && \text{Loss Weighted,}\\
\lambda_i\left(t\right) &= \frac{\exp\left(\beta\left(\frac{s_i}{\sum_{l=1}^{m}\left|s_j\right|}\right)(t)\right)}{\sum_{j=1}^{k}\exp\left(\beta\left(\frac{s_i}{\sum_{l=1}^{m}\left|s_j\right|}\right)(t)\right)}, i=1,..,k && \text{Loss Normalized.}
\end{aligned} \tag{3.5}$$

In eq. 3.5 the $s_i\left(t\right)$ is the recent rate of change of the i^{th} loss term (*e.g.* $s_i\left(t\right)=\mathcal{L}_i\left(t\right)-\mathcal{L}_i\left(t-1\right)$ or a more accurate finite difference approximation), and β is a tunable hyper-parameter. If $\beta>0$ is used, SoftAdapt will assign more weight to the worst performing component of the loss function (*i.e.* the component with most positive rate of change), $\beta<0$ favours the best-performing losses (most negative rate of change) and by using $\beta=0$ gives equal weights (classic SoftAdapt method). The default value of the β hyperparameter according to the authors is 0.1. In the normalized version of the SoftAdapt the $s_i\left(t\right)$ is normalized before using it.

3.4 Statistical analysis of the evaluation results

In order to evaluate and compare the efficacy of various self-adaptation approaches for balancing the coefficients of loss terms, a systematic procedure was employed. Multiple models were trained for each self-adaptation approach using different initial conditions, defined by varying seed values. This resulted in the creation of distinct populations of models for each self-adaptation approach.

Each model was then evaluated based on statistical parameters described in Section 3.2. These parameters serve as a quantitative measure of the model's performance, capturing various characteristics of their behaviour.

Following the evaluation of individual models, a comparative analysis of the populations of models was conducted. This involved performing a Kruskal-Wallis test on the statistical parameters of each model population.

The Kruskal-Wallis test, a non-parametric statistical test, is used to compare the medians of two or more independent groups. It serves as a non-parametric equivalent to the one-way analysis of variance (ANOVA), but unlike ANOVA, it does not assume a normal distribution of data, making it a more flexible choice when dealing with data distributions that deviate from normality. However, the test does require that samples within groups are independent and identically distributed.

The null hypothesis of the Kruskal-Wallis test posits that all samples originate from the same distribution or from distributions with the same median. Should the p-value yielded by the test fall below the significance level (typically 0.05), the null hypothesis is rejected. This indicates a significant difference between at least two of the groups.

After identifying the statistical parameters that differ significantly between models using the Kruskal-Wallis test, a post hoc test was conducted to discern which models differ and to what extent. This helps in establishing a ranking among the different approaches.

The post hoc test employed in this study is Tukey's range test, also known as Tukey's Honest Significant Difference (HSD) test. This statistical method, often paired with ANOVA, helps identify means that are significantly different from each other. While ANOVA determines if a significant difference exists among groups, it does not specify which groups differ significantly. This is where Tukey's range test becomes instrumental. It compares all possible pairs of means, particularly useful when variances are assumed equal across groups.

4 Experiments and results

The research objective is to establish a robust and consistent evaluation process for various approaches to the self-adaptation of loss component weights in the GCNN model, as it is outlined in Section 1. The primary goal is to answer the scientific question posed in the study by investigating whether the proposed method, as described in Section 3.1, can fulfil the set target. To achieve this, a sequence of experiments was conducted. Initially, an optimized model with a constant linear scalarization loss coefficients balancing method was utilized to assess its transferability. The results of these experiments are presented in Section 4.1. Subsequently, different self-balancing methods were implemented during the training process of the model, and their performance was evaluated using the proposed method. The evaluation method incorporates specific statistical parameters, as defined in Section **??**, to gauge the performance of each approach. The overall aim of the research is to comprehensively evaluate the effectiveness of various loss coefficient self-balance methods in the GCNN model. By performing a series of experiments and employing the proposed evaluation process, the study seeks to provide a robust and consistent assessment of the performance of each method.

4.1 Model transferability tests

Model transferability refers to the ability of a model trained on atomistic molecular simulation results to be effectively utilized in CG simulations under different conditions. Experiments targeting the evaluation of the transferability of the model when applied to CG simulations at temperatures different from those in the training dataset were performed. The study consists of two main stages. Firstly, the performance of a model trained at a specific temperature is assessed in CG simulations conducted at various temperatures. This evaluation allows for an understanding of how well the model performs when extrapolated to different temperature regimes. The results of this stage are presented in Section 4.1.1. Following that, the study investigates whether the hyperparameters of the loss function (as described in Section 3.1), which have been optimized for the training process using a dataset of a specific temperature, can be successfully applied to the training process using a dataset of a different temperature. This analysis aims to determine if the hyperparameters can be effectively transferred and used in a generalized manner. The findings of this investigation are presented in Section 4.1.2. By conducting these stages of analysis, the research sheds light on the model's transferability in CG simulations across different temperature conditions and explores the applicability of optimized hyperparameters in training processes with varying temperature datasets. The results provide valuable insights into the model's robustness and potential for generalization.

4.1.1 Temperature Transferability

Initially, the temperature transferability of a trained model was evaluated. Two routes of evaluation were followed. In the first one, a model trained using a dataset derived from the atomistic molecular dynamic simulation at 300 K was used for CG simulations at temperatures ranging between 280 and 340 K. In the second one, the influence of the training temperature in the CG simulation at 340 K was investigated.

The simulation results indicate that the trained model produces consistent results when it is used for CG simulations at different temperatures to the one of the training. The pair correlation functions, $g(r)$, calculated from the simulation results at different temperatures are presented in Figure 7a. Their lineshapes are approximately the same, and this is also indicated

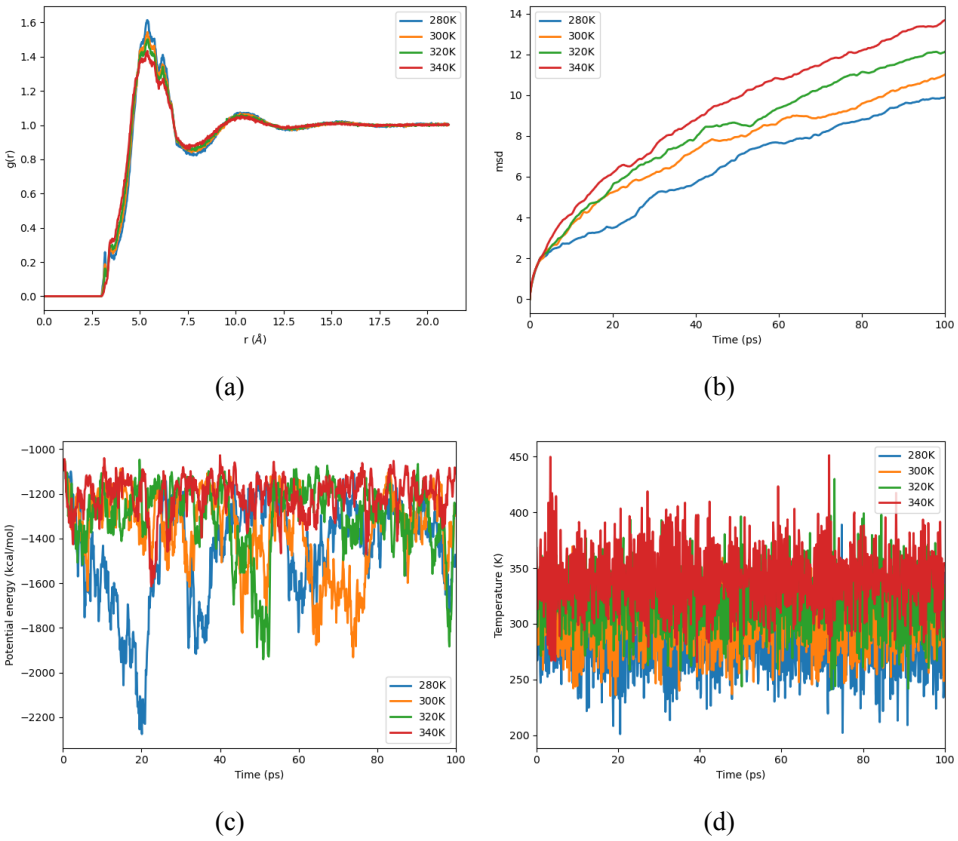
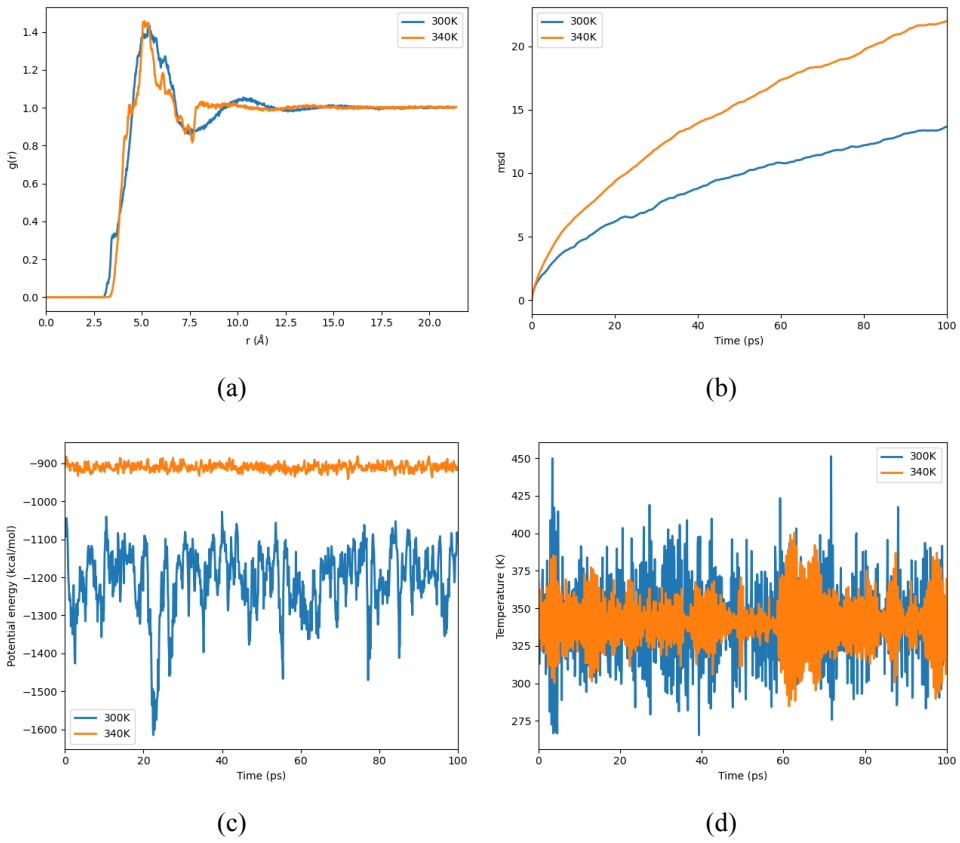


Figure 7: Results of the coarse-grained simulations at temperature ranged between 280K and 340K using a model trained at 300K. (a) Radial distribution function. (b) Mean square displacement. (c) Potential energy, with a red line the mean potential energy from atomistic simulation is indicated. (d) Temperature.

by the p-value of the Kruskal-Wallis test, which is 0.9955. Although, two small peaks at approximately 3.2 and 3.5 Å, that are not expected, are presented in all cases. The intensity of the peaks decreases as the simulation temperature increases. The expected behaviour is also presented in the mean square displacement (Figure 7b) with an increase in the simulation temperature leading to higher values; particles' mobility increases with increased temperature. In the case of coarse-grained simulated potential energy (Figure 7c) it has a lower fluctuation with the increase of simulation temperature; a lower standard deviation is observed as the simulation temperature increases. Specifically, the mean value of the coarse-grained simulated potential energy is -1452.5(±231.3) *kcal/mol*, -1365.0(±177.3) *kcal/mol*, -1300.2(±159.4) *kcal/mol*, and -1207.0(±87.5) *kcal/mol* for simulation at 280K, 300K, 320K, and 340K, respectively. The mean total potential energy from atomistic simulations for 300K and 340K are -1099.7(±17.3) *kcal/mol*, and -906.2(±18.5) *kcal/mol*, respectively. Comparing these with the mean total energy from coarse-grained simulations at the relative temperatures shows that in both cases the latter are significantly lower. Finally, the coarse-grained simulated temperatures are 279.8(±25.3)K, 299.91(±24.85)K, 319.6(±26.6)K, and 340.4(±27.3)K, which are very close to the targeted ones and with approximately same fluctuation.

The influence of the training temperature in the simulation results was also evaluated by comparing the simulation results using a model trained at the same and different temperatures. Specifically, trained models at 300K and 340K were used for simulations at 340K. The hyperparameters in the training of both models were the same.



(a) (b)

(c) (d)

Figure 8: Results of the coarse-grained simulations at 340K using models trained at 300K and 340K. (a) Radial distribution function. (b) Mean square displacement. (c) Potential energy. (d) Temperature.

In Figure 8 the results of the coarse-grained simulations at 340K using models trained at 300K and 340K are presented. The radial distribution function (Figure 8a) presents a smoother lineshape when the model trained in the lower temperature was used, although there is a small shoulder at approximately 3 Å that vanished when the model trained at 340K was used. The training temperature seems to also affect the simulated mean square displacement (Figure 8b), which should be the same as the simulation temperature is the same, in both cases. Using a trained at 300 K model results in lower mean square displacement over time compared to that when a model trained at 340K was used. In the case of coarse-grained simulated potential energy (Figure 8c) using a model trained at 340 K results in to mean value closer to that of the atomistic simulations and with significantly decreased fluctuation. Specifically, the mean value of the total potential energy from atomistic simulations for 340K is -906.2(±18.5) kcal/mol whereas the coarse-grained simulated total potential energy is -1207.0(±87.5) kcal/mol and -910.0(±9.1) kcal/mol, for models trained at 300 and 340 K, respectively. Finally, in both cases, the coarse-grained simulation temperature is close to the targeted one although using the model trained at 300 K results in higher fluctuation compared to when the model trained at 340K is used. The coarse-grained temperature when the model is trained at 300K is 340.4(±27.3)K and when the model is trained at 340K is 340.0(±17.6)K.

4.1.2 Loss hyperparameters indicative tests

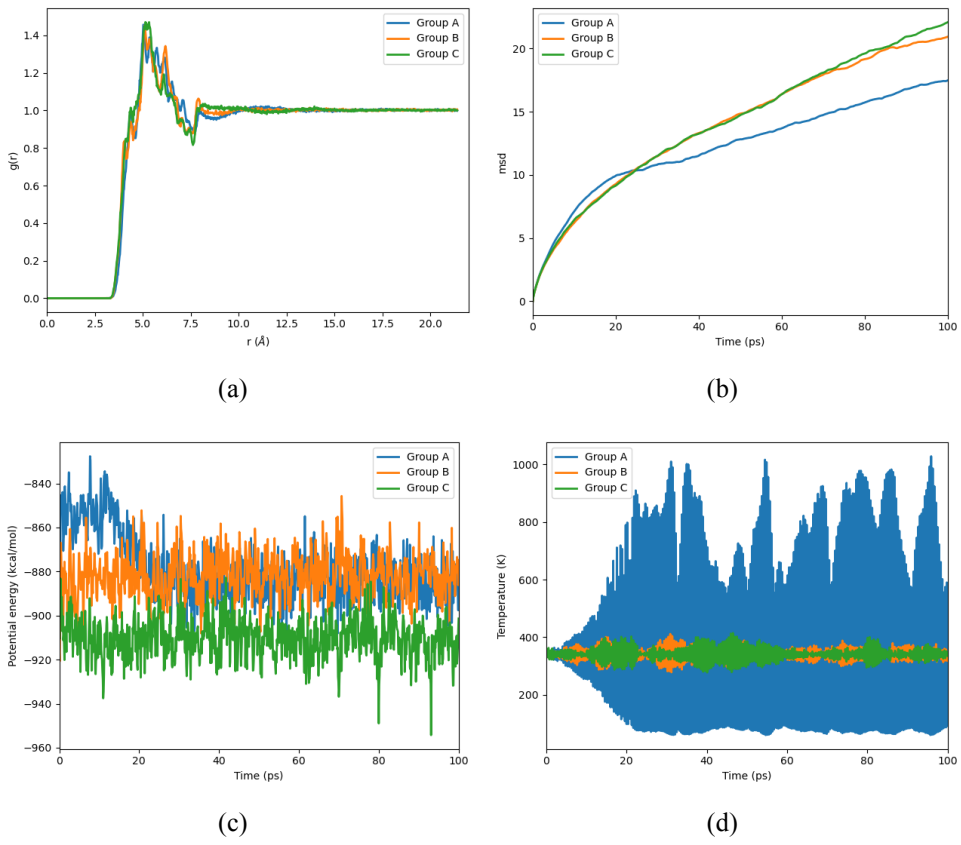


Figure 9: Results of the coarse-grained simulations at 340K using models trained at 300K and 340K. (a) Radial distribution function. (b) Mean square displacement. (c) Potential energy. (d) Temperature.

As it has been mentioned in Section 3.1 the model has three hyperparameters included in the

loss function, the distance and the exponent parameters in the excluded volume (eq. 3.2), and the coefficient λ of the energy loss component. Three sets of hyperparameters that are related to the loss function had been optimized for training and simulation at 300 K by grid search of a high hyperparameter space in previous research efforts. Their use for training a model at 340 K was evaluated by performing experiments using them. The loss-related hyperparameter sets are presented in Table 1

Table 1: Optimized hyperparameter sets evaluated for training models at 340K

	σ_{ex}	n_{ex}	λ
Group A	5	5	1
Group B	5	6	1
Group C	5	7	0.1

The simulation results using the models trained at 340K and using the three different hyperparameter sets are presented in Figure 9. All three sets of hyperparameters result to pair correlation functions with the same lineshape (Figure 9a). Although, the hyperparameters of Group A result in high fluctuation of temperature, a significant decrease in the potential energy after approximately 20 ps, and not expected behaviour of the MSD curve. Between the hyperparameters of Group B and Group C, the latter presents potential energy closer to that calculated from atomistic simulations and a lower fluctuation in the temperature. The MSD curve is approximately the same for Group B and Group C.

4.2 Loss components balancing methods

This section presents the results of the investigation into the performance of the proposed method for evaluating different approaches to self-balancing loss coefficients. The experiments conducted for this purpose involved the implementation of various self-balancing approaches, including linear scalarization, ratio, and normalization of loss components. Additionally, the experiments considered the maximum values of the loss components from the current epoch and the values from the 5, 10, and 100 previous epochs. Furthermore, three versions of SoftAdapt were also tested. Multiple experiments were performed for each approach, starting from different initial conditions (seed values). To evaluate the performance of each approach, the statistical parameters described in Section3.2 were calculated based on the experimental results. These statistical parameters were then utilized in the proposed evaluation method, as outlined in Section 3.4. By conducting these experiments and utilizing the proposed evaluation method, the study provides insights into the effectiveness of different self-balancing approaches for loss coefficients. The statistical parameters derived from the results contribute to a comprehensive evaluation and comparison of each approach's performance.

4.2.1 Constant Loss methods

The physical parameters of the system simulated from models trained using predefined self-adaptation methods are displayed in Figure 10. As shown, when the maximum value of the loss terms is utilized for calculating the coefficients of the loss terms, the simulation results exhibit high fluctuations in the case of both CG simulated potential energy and temperature (Figure 10c and d), while the slope of their MSD function (Figure 10b) decreases. However, in the case of the ratio approach, the simulated parameters are consistent with those obtained using optimized linear scalarization loss term coefficients.

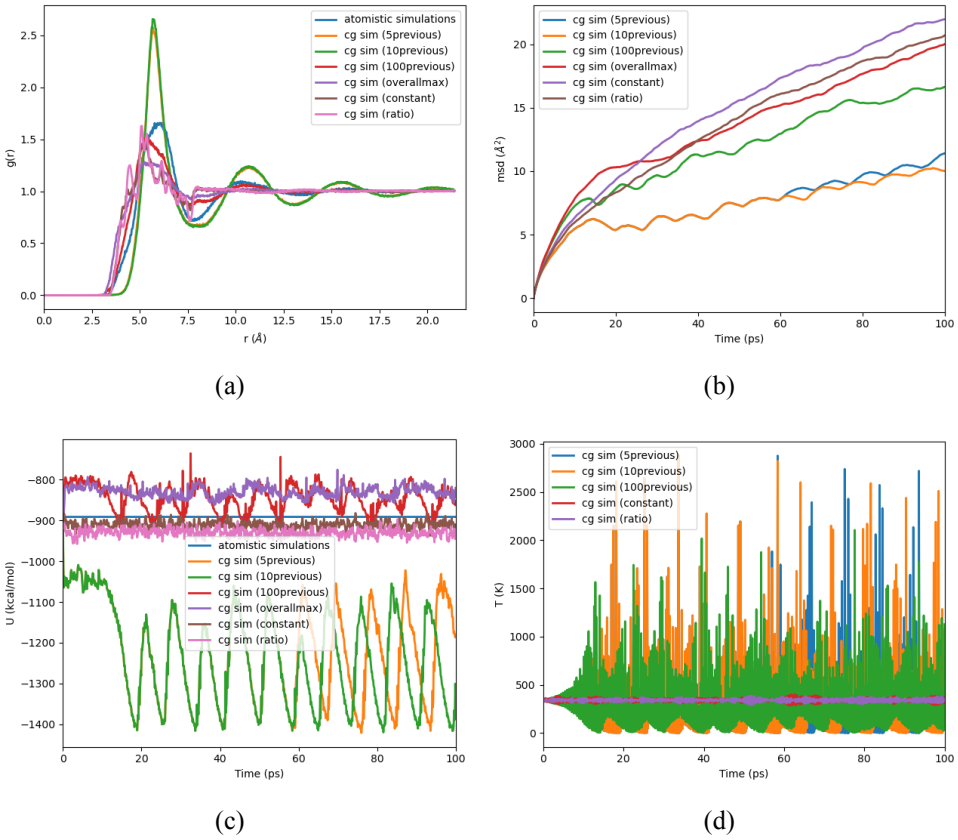


Figure 10: Results of the coarse-grained simulations using predefined self-balancing approaches. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

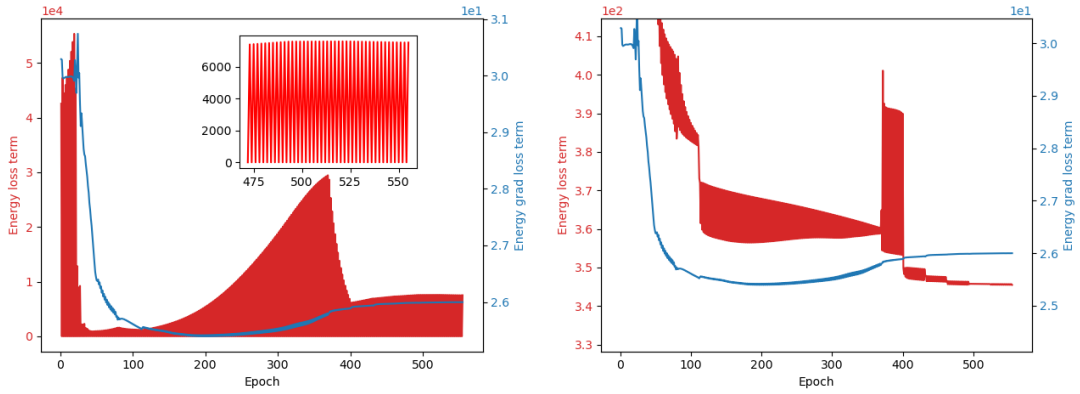
These observations are further supported by the statistical parameters derived from the CG simulation results, as described in Section 3.2. As demonstrated in Table 2, the ratio method yields results comparable to those obtained with optimized linear scalarization one with constant weights, except for the number of epochs needed for the model's complete training. Conversely, in all instances where the maximum value of loss terms was employed for balancing the loss terms, high CV values are calculated for both the CG simulated potential energy and temperature.

The evolution of the loss terms during the model training using the ratio method provides insights into the mechanism of balancing the loss terms. As shown in Figure 11a, the balanced energy loss term exhibits unusual behaviour, with an initial sharp decrease in the first few epochs, followed by a gradual increase, and then another sharp decrease. A clearer understanding of the balanced energy loss term's evolution is provided by the inset of Figure 11a, which displays the last 10 % of training epochs in a magnified view. It becomes evident that the balanced energy loss term oscillates between a high value and approximately zero, suggesting that the model is optimized for one of the loss components in each epoch. In epochs where the balanced energy loss term has a high value, it has a greater influence on the total loss compared to the energy grad loss term. Conversely, when its value reaches approximately zero, the energy grad loss term dominates the loss function. The progression of the unbalanced loss terms

Table 2: Statistical parameters extracted from the training and simulation of models using intuitive approaches for the self-adaptation of the loss terms coefficients.

Name	N_{ep}	$<t_{ep}>$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	$<T>$ (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
5previous	626	128.4	1.0	34.5	9.9	339.7	128.7	0.16	$1.4{\cdot}10^{-11}$	0.073
10previous	624	128.6	0.9	36.2	9.6	338.6	132.5	0.17	$5.0{\cdot}10^{-13}$	0.061
100previous	365	128.3	1.0	6.5	3.9	336.4	100.4	0.08	$2.8{\cdot}10^{-3}$	0.1
overallmax	377	128.7	1.0	8.4	1.6	341	85.2	0.153	$1.3{\cdot}10^{-10}$	0.13
ratio	555	128.7	0.8	2.3	1.0	340.1	4.7	0.144	$1.9{\cdot}10^{-9}$	0.145
constant	375	128.1	0.9	0.5	1.0	340.0	5.2	0.172	$2.5{\cdot}10^{-13}$	0.143

indicates that the model's learning ability increases throughout the training period, as their sum consistently decreases.



(a) (b)

Figure 11: Evolution of (a) balanced and (b) unbalanced loss components during the training of the model using the ratio approach.

4.2.2 Efficiency of SoftAdapt and optimum predefined methods

Based on the findings discussed in Section 4.2.1, it can be concluded that among the predefined self-adaptation methods, only the ratio approach yielded CG simulation results as good as the optimized linear scalarization method with constant weights which remains the best-performing approach in relation the prediction of properties in comparison to the atomistic simulations ground truth. To further investigate the performance of the ratio approach, it was compared with the three variations of the SoftAdapt method and the constant weight linear scalarization method by conducting multiple experiments with different initial states, specifically varying initialization seed numbers. This approach allowed for the development of a population for each method, thereby increasing the confidence in the derived statistical parameters for each technique. Five different seeds were utilized in the experiments, with each seed being shared across all the methods under comparison.

Optimized linear scalarization constant weights approach Figure 12 displays the benzene liquid properties calculated from the simulation results, using models trained by employing the

constant weight approach. As shown in Figure 12a, the pair correlation function's lineshape is almost identical for all seed values, except for the fourth seed. In this particular case, the pair correlation function exhibits a peak at approximately 3 Å, which does not have a clear physical explanation. Additionally, in all instances, the pair correlation function's peak appears at lower r values compared to those from atomistic simulations.

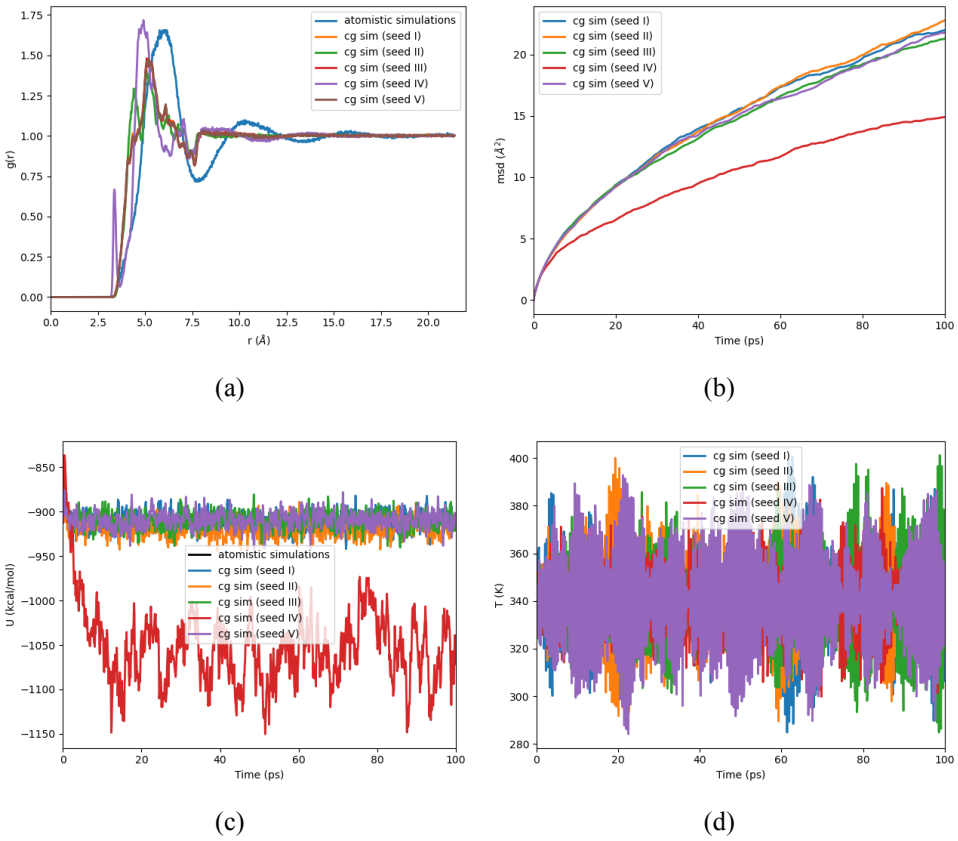


Figure 12: Results of the coarse-grained simulations using the optimized constant weight approach. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

The pair correlation functions do not exhibit the same smoothness as those from atomistic simulations, and they lack the damped oscillatory form observed in the reference function for r values higher than 7.5 Å. In terms of the MSD function, the model trained with the fourth seed demonstrates behaviour resembling a frozen system, which is distinct from all other cases, as depicted in Figure 12b. The failure of the model trained with the fourth seed is also apparent in the CG potential energy; while all other models result in potential energy values close to those of atomistic simulations and remain stable over time, the fourth seed model produces a lower simulated potential energy with significant oscillations throughout the simulation period. Ultimately, the simulated temperature of the system oscillates around the target value for all cases.

The behaviours illustrated in the diagrams of Figure 12 can be quantified using the statistical parameters described in Section 3.2 (Figure 3. The absolute percentage difference between the mean potential energy from atomistic and CG simulations ranges between 0.36% and 1.44%

for all cases, except for the fourth seed, where the difference increases to approximately 16%. The $CV_{U_{CG}}$ also rises from 1% to 4.1%. Conversely, the mean temperature remains close to 340 K in all cases, with low fluctuations (CV_T of approximately 5%). The K-S test indicates that none of the models results in a pair correlation function that is statistically similar to that from atomistic simulation. Lastly, all models require approximately 400 training epochs, with each epoch lasting 129 seconds.

Table 3: Statistical parameters extracted from the training and simulation of models using the optimized constant weight approach for the self-adaptation of the loss terms coefficients for different seed values.

	N_{ep}	$<t_{ep}>$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	<T> (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
seed I	375	129.1	0.9	0.46	1.0	340.0	5.2	0.17	$2.48 \cdot 10^{-13}$	0.143
seed II	416	128.5	0.9	1.44	1.0	340.1	5.3	0.23	$5.3 \cdot 10^{-24}$	0.154
seed III	375	129.1	0.9	0.45	1.0	340.0	5.5	0.18	$4.3 \cdot 10^{-14}$	0.143
seed IV	415	127.9	1.0	16.09	4.1	340.0	4.2	0.11	$1.69 \cdot 10^{-5}$	0.098
seed V	375	129.1	0.8	0.36	1.0	340.0	6.3	0.19	$1.08 \cdot 10^{-15}$	0.140

Ratio approach The CG-simulated physical parameters using models where the ratio approach is employed for self-balancing of the loss terms are presented in Figure 13. The pair correlation functions derived from these models are strikingly similar to those derived from models trained using the optimized constant weight approach. In this case, the pair correlation functions lack the damped oscillatory behaviour observed in the atomistic simulation for *r* values above 7.5 Å and exhibit multiple peaks instead of a smooth lineshape.

Regarding the MSD, all models except the one of seed IV exhibit the expected behaviour. The latter indicates a frozen system. Interestingly, this is the same seed that yields peculiar results in the optimized constant weight approach. The failure of the seed IV model is also apparent in the CG simulated potential energy and temperature. For this model, the CG simulated potential energy is significantly lower than the atomistic simulated potential energy, and the temperature exhibits high oscillations. In contrast, for all other models, the CG simulated potential energy is significantly closer to that of the atomistic simulated potential energy, with low oscillations.

The statistical parameters extracted from these models also indicate similar behaviour (Table 4). All models, except for the seed IV model, present an absolute percentage difference between the mean CG and atomistic simulated potential energy of approximately 2.5%. For the seed IV model, this parameter is 45.3%. Using seed IV also results in an increase of $CV_{U_{CG}}$ to 4.8% from approximately 1%. Moreover, it leads to an increase in the oscillation of temperature over simulation time from approximately 5% to 87%. The dissimilarity between the CG and atomistic simulated pair correlation functions is also highlighted by the significantly low p-value of the K-S test. Lastly, training the model using the ratio approach for self-balancing of the loss terms requires between 364 and 555 epochs of approximately 128 seconds each.

SoftAdapt Original Variant Figure 14 displays the CG-simulated parameters from models employing the original variant of the SoftAdapt approach. The lineshapes of the CG-simulated pair correlation functions differ significantly from those of the atomistic simulations. This difference is also highlighted by the low p-value of the K-S test (Table 5). Conversely, the line-

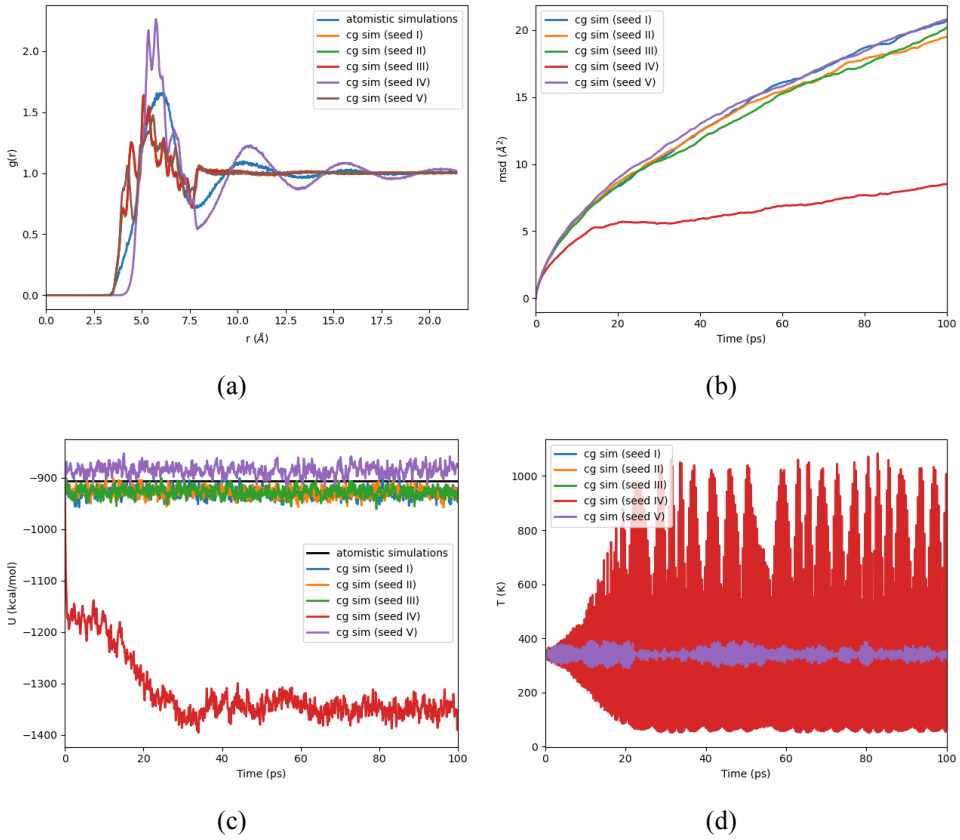


Figure 13: Results of the coarse-grained simulations using the ratio approach. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

Table 4: Statistical parameters extracted from the training and simulation of models using the ratio approach for the self-adaptation of the loss terms coefficients for different seed values.

	N_{ep}	$\langle t_{ep} \rangle$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	$\langle T \rangle$ (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
seed I	555	128.7	0.8	2.6	1.0	340.1	4.7	0.144	$1.86 \cdot 10^{-9}$	0.145
seed II	555	128.7	0.8	2.4	1.0	340.2	5.2	0.164	$3.74 \cdot 10^{-12}$	0.124
seed III	554	128.7	0.8	2.4	1.0	340.2	4.8	0.163	$5.21 \cdot 10^{-12}$	0.139
seed IV	312	128.1	1.2	45.3	4.8	339.9	87.0	0.149	$4.25 \cdot 10^{-10}$	0.043
seed V	364	128.4	1.0	2.2	1.2	339.9	5.9	0.178	$2.98 \cdot 10^{-14}$	0.136

shapes of the MSD functions exhibit the expected form (Figure 14b), and the slope of their last 30% of timesteps closely aligns with the results from models using optimized constant weights.

Moreover, the CG-simulated potential energy remains relatively constant for all seeds, although their mean value is lower than that of atomistic simulations. As shown in Table 5, the absolute percentage difference ranges between 1.7% and 10.7%. The CG-simulated temperature has a mean value of 340K for all cases, with an oscillation of approximately 5%. The models require around 600 epochs (between 557 and 638) with a duration of 116 seconds each.

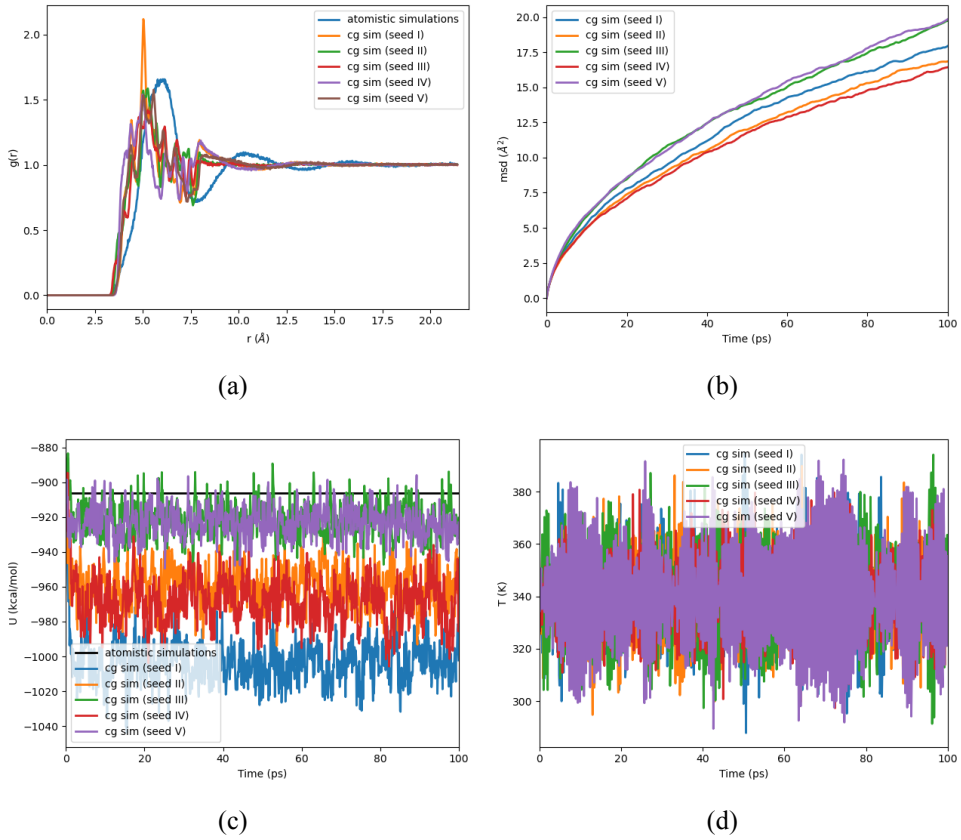


Figure 14: Results of the coarse-grained simulations using the original variant of SoftAdapt approach. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

Table 5: Statistical parameters extracted from the training and simulation of models using the original variant of SoftAdapt approach for the self-adaptation of the loss terms coefficients for different seed values.

	N_{ep}	$<t_{ep}>$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	$<T>$ (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
seed I	580	116.1	0.5	10.7	1.3	340.0	4.5	0.103	$4.87 \cdot 10^{-5}$	0.116
seed II	637	117.5	0.5	5.9	1.2	339.9	4.3	0.186	$1.57 \cdot 10^{-15}$	0.111
seed III	638	115.3	0.4	1.7	1.1	339.9	4.9	0.186	$1.57 \cdot 10^{-15}$	0.124
seed IV	557	115.9	0.6	6.8	1.4	340.2	3.8	0.122	$6.67 \cdot 10^{-7}$	0.105
seed V	580	116.2	0.6	1.9	0.9	339.9	5.9	0.127	$1.91 \cdot 10^{-7}$	0.129

Weighted SoftAdapt Figure 15 displays the CG-simulated physical parameters derived from models where the weighted variation of SoftAdapt has been employed for self-balancing the loss term coefficients. The CG pair correlation functions show significant differences compared to the atomistic simulations (Figure 15a), which is also indicated by the low p-values of the K-S tests (Table 6).

The MSD function lineshapes have the expected form, and the slopes of their latter parts are approximately the same as those in the optimized constant weight case. Furthermore, while the

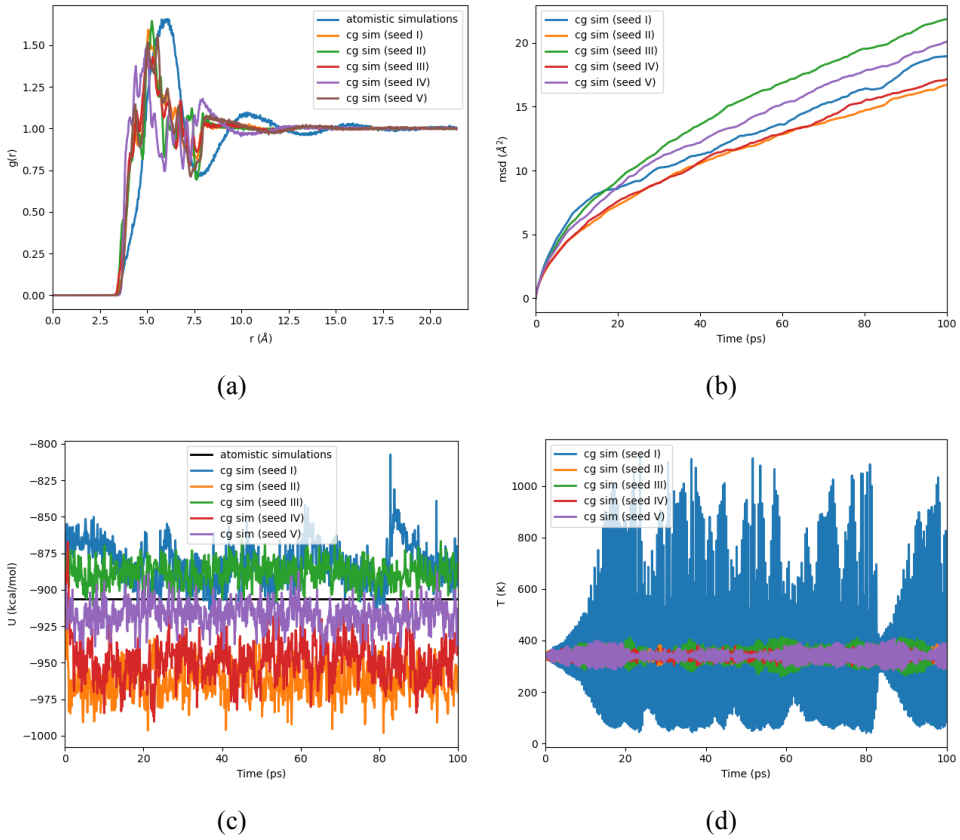


Figure 15: Results of the coarse-grained simulations using the weighted SoftAdapt approach. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

CG potential energy exhibits low oscillation in all cases, with a coefficient of variance lower than 1.6% in each instance, the mean values are slightly different.

A good agreement between the mean simulated temperature and the target value is observed, with low oscillation in all cases, except for the seed I case where the simulated temperature exhibits high oscillation over the simulation time (Figure 15). Lastly, the number of epochs needed ranges from 550 to 1500, with four out of five models requiring fewer than 725 epochs.

Normalized SoftAdapt The CG-simulated physical parameters using models with the normalized SoftAdapt approach indicate that this method leads to worse results than the original variant of SoftAdapt. The CG-simulated pair correlation functions show a significant difference compared to those from atomistic simulations, as illustrated in Figure 16a and Table 8. Moreover, the lineshape of most CG-simulated MSD functions exhibits low slope values (16b).

While the simulated temperature of the system has a mean value close to 340K, it displays high oscillation for all cases, with its coefficient of variation being higher than 46%, except for the seed IV model, which presents a small oscillation and a coefficient of variation of 4.5%. Conversely, the CG potential energy mean values are close to the atomistic simulated ones in

Table 6: Statistical parameters extracted from the training and simulation of models using the weighted SoftAdapt approach for the self-adaptation of the loss terms coefficients for different seed values.

	N_{ep}	$<t_{ep}>$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	$<T>$ (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
seed I	618	116.2	0.6	2.6	1.6	341.3	80.4	0.171	$3.51 \cdot 10^{-13}$	0.129
seed II	725	115.7	0.8	6.2	1.2	340.0	4.1	0.182	$6.95 \cdot 10^{-15}$	0.105
seed III	1500	115.1	0.7	1.9	0.9	339.9	9.4	0.164	$3.74 \cdot 10^{-12}$	0.141
seed IV	568	115.2	0.7	4.7	1.5	339.7	4.0	0.137	$1.34 \cdot 10^{-8}$	0.115
seed V	550	116.7	0.7	1.2	1.0	340.0	7.0	0.13	$8.80 \cdot 10^{-8}$	0.131

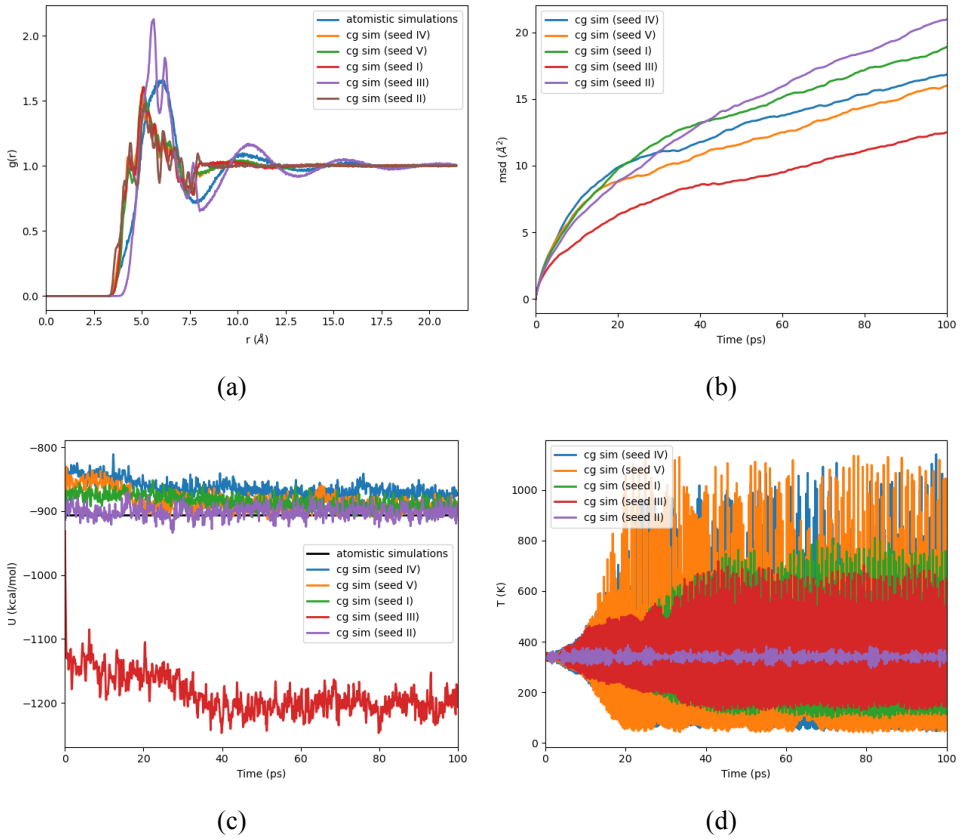


Figure 16: Results of the coarse-grained simulations using the normalized SoftAdapt approach. (a) Pair correlation function, the pair correlation function from the atomistic simulation is presented with a blue line. (b) Mean Square Displacement. (c) Potential Energy, the mean potential energy of the atomistic simulations is presented with a blue line. (d) Temperature, the targeted temperature of the system was 340 K.

all cases, except for the seed II case, where a lower value of approximately 31% is observed.

The models also require significantly more epochs for training, with four out of five cases needing more than 1000 epochs.

Table 7: Statistical parameters extracted from the training and simulation of models using the normalized SoftAdapt approach for the self-adaptation of the loss terms coefficients for different seed values.

	N_{ep}	$<t_{ep}>$ (s)	$\sigma_{t_{ep}}$ (s)	ΔU (%)	$CV_{U_{CG}}$ (%)	$<T>$ (K)	CV_T (%)	K-S D	K-S p-value	MSD slope
seed I	1449	115.3	0.7	2.3	1.2	339.8	53.9	0.152	$1.71 \cdot 10^{-10}$	0.097
seed II	610	116.0	0.5	0.5	1.1	340.0	4.5	0.191	$2.33 \cdot 10^{-16}$	0.137
seed III	1497	115.4	0.6	31.0	2.5	340.0	46.9	0.099	$1.1 \cdot 10^{-4}$	0.069
seed IV	1302	116.2	0.7	4.6	1.5	340.8	84.3	0.134	$3.04 \cdot 10^{-8}$	0.085
seed V	1083	115.9	0.6	2.3	1.7	341.3	91.0	0.122	$6.67 \cdot 10^{-7}$	0.089

4.2.3 Methods performance and technical comparison

In the context of evaluating self-balancing approaches, box plots of different statistical parameters play a critical role. A box plot, also known as a box-and-whisker plot, is a graphical method that provides a five-number summary of a dataset. This summary includes the minimum, the first quartile (Q1), the median, the third quartile (Q3), and the maximum. The "box" in a box plot represents the interquartile range, that is, the distance between Q1 and Q3, thus enclosing the middle 50% of the data. The line within the box shows the median of the dataset. The "whiskers" that extend from the box denote the variability beyond the lower and upper quartiles, thereby providing a complete view of the dispersion of the data. Outliers, if present, are usually depicted as individual points lying beyond the whiskers. Box plots can be drawn either vertically or horizontally and are particularly useful for comparing one or more datasets. They are essential tools for identifying outliers, measuring variability, and detecting symmetry in the data, and they offer considerable value when comparing samples from different populations. Despite the limited populations within each approach, certain preliminary conclusions can be derived.

As illustrated in Figure 17, there appears to be a noticeable difference in the required number of epochs for model training across varying self-balancing approaches. The constant weight approach, for instance, demands the least number of epochs, while the normalized version of SoftAdapt requires the most.

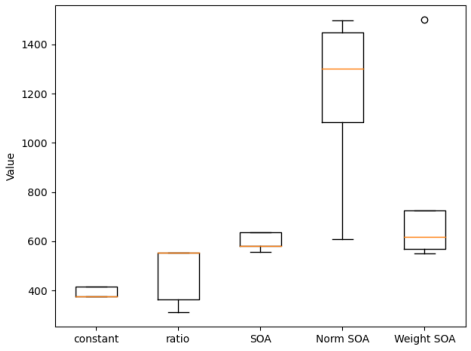
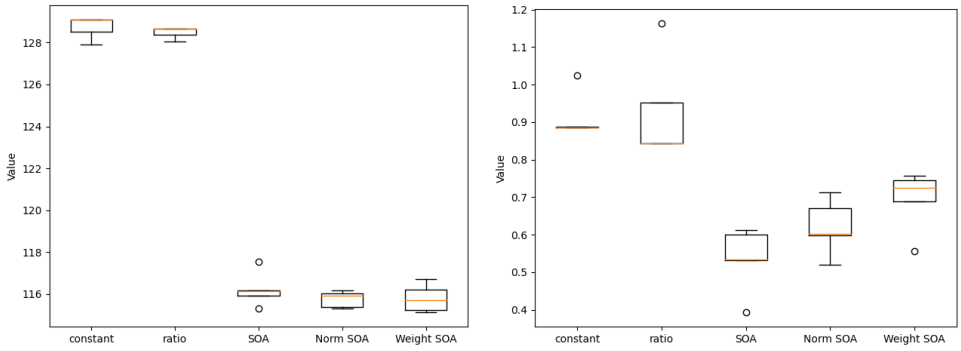


Figure 17: Box plots of the number of epochs for the linear scalarization constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt..

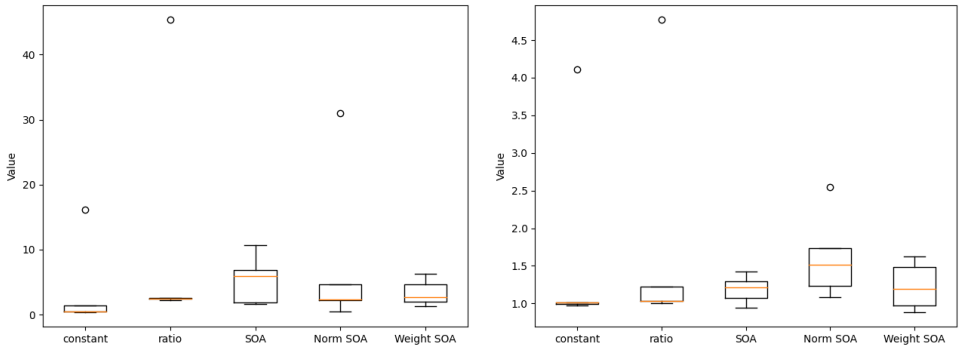
The self-balancing methods incorporated within the model seem to influence not only the duration but also the stability of the training epochs (Figures 19a and b). All the SoftAdapt vari-



(a) (b)

Figure 18: Box plots of the (a) mean and (b) standard deviation of the duration of training epoch for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.

ants require less time per epoch, and the time required is notably more consistent compared to the constant weight and ratio approaches. This difference of approximately 10 seconds attempts to offset the larger number of epochs needed for the SoftAdapt methods.

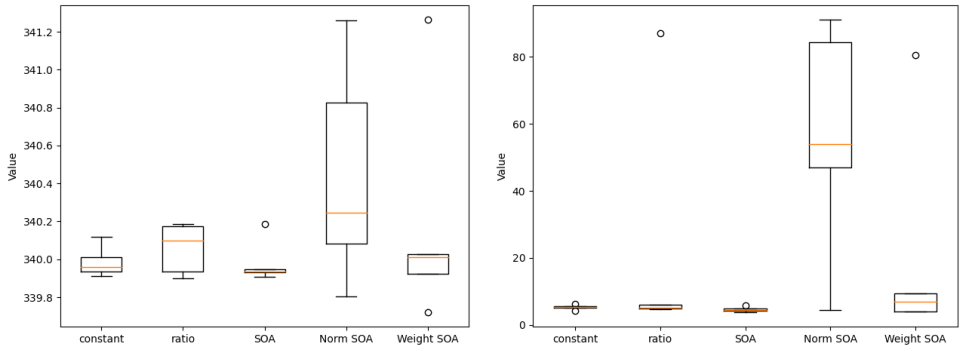


(a) (b)

Figure 19: Box plots of the (a) absolute percentage difference between the CG and atomistic simulated potential energy and (b) its coefficient of variation for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.

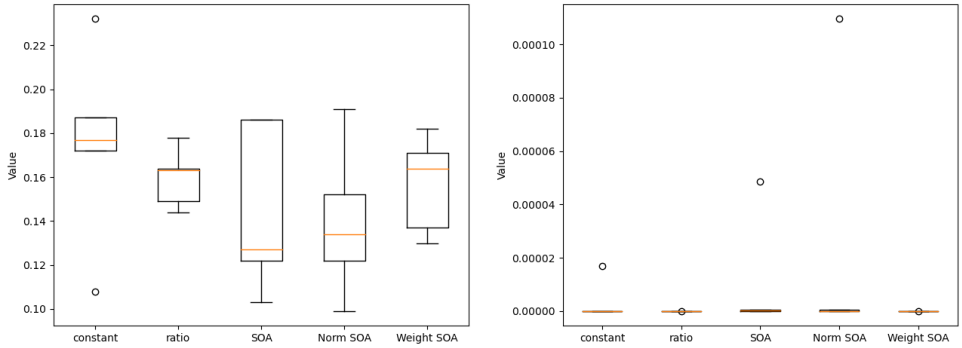
When considering the absolute difference between the mean CG and atomistic simulated potential energies, no substantial difference is discerned between the approaches (Figure 19a). The constant weight and ratio approaches appear to have a lower mean value compared to the SoftAdapt methods, but the presence of an outlier in both cases, coupled with a small population size, complicates the drawing of clear conclusions. This complexity also surfaces in the case of the $CV_{U_{CG}}$, where despite the lower mean value and dispersion for the constant weight and ratio approaches, outliers are present in both instances of cloud interpretation (Figure 19b).

All methods deliver a simulated temperature close to the expected value, with similar dispersion if outliers are accounted for (Figure 20a). However, the box plots of the CV_T parameter indicate that the normalized version of SoftAdapt exhibits significantly higher oscillations in the simulated temperature relative to the other methods (Figure 20b).



(a) (b)

Figure 20: Box plots of the (a) mean CG simulated temperature and (b) its coefficient of variation for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.



(a) (b)

Figure 21: Box plots of the (a) statistic and (b) p-value of K-S test for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.

In the case of the K-S test results, no notable difference between the self-balancing methods is observed. Both the statistic and p-value parameters display similar behavior across the various approaches (Figure 21). Lastly, a significant difference between the self-balancing methods emerges in the MSD slope parameter. The constant weight approach results in the highest MSD slope, while the normalized version of SoftAdapt yields the lowest.

4.2.4 Post hoc analysis

A robust, repeatable, and quantifiable comparison of the performance of various self-balancing methods, as against the boxplots used in Section 4.2.3, is achieved using the methodology outlined in Section 3.4. This approach employs the Kruskal-Wallis test, a non-parametric statistical test, to identify statistical parameters where at least one approach exhibits a distinct behavior compared to the others. Subsequently, Tukey's test, a post-hoc analysis, is utilized to pinpoint the differing methods and quantify the extent of these differences.

The results derived from the Kruskal-Wallis test are encapsulated in Table 5. If we adopt a more lenient p-value of 0.1, it becomes evident that the null hypothesis of the Kruskal-Wallis

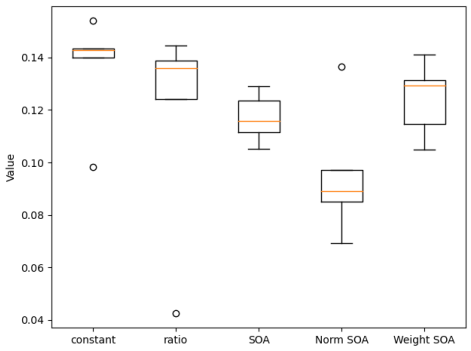


Figure 22: Box plots of the slope of the last 30% of the MSD function for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.

test is dismissed for several parameters. These include the total number of epochs, the mean iteration time, the standard deviation of iteration time, and the MSD slope parameters.

This result implies that, for these specific parameters, the mean value of at least one method is statistically significantly different from the others. This finding aligns with the observations made from the box plots of the statistical parameters presented in Section 4.2.3.

By employing this two-step statistical process, the Kruskal-Wallis test followed by Tukey's test, we are better equipped to discern subtle differences between methods and obtain a more nuanced understanding of the comparative performance of different self-balancing approaches.

Table 8: Results of Kruskal-Wallis test for the different statistical parameters.

	N_{ep}	$<t_{ep}>$	$\sigma_{t_{ep}}$	ΔU	$CV_{U_{CG}}$	$<T>$	CV_T	K-S D	K-S p-value	MSD slope
statistic	17.78	17.79	19.61	4.45	4.31	2.80	6.57	2.88	2.88	8.37
p-value	0.0014	0.0014	0.0006	0.35	0.35	0.59	0.16	0.58	0.58	0.079

The limited number of experiments carried out for each approach, along with the simulation results of potential energy and temperature discussed in Section 4.2, suggest a higher threshold should be adopted when selecting parameters for Tukey's test. Therefore, a value of 0.35 is used in this instance, extending the scope of Tukey's test to the absolute difference between the mean CG and atomistic simulated potential energies, and the coefficient of variation for the CG simulated potential energy and temperature.

The results of the pairwise Tukey's test for these parameters are compiled in Table 10. The "rejected" column in the table corresponds to the null hypothesis' rejection for this test, again utilizing a significance level of 0.35.

Leveraging the outcomes from Tukey's test, a ranking system can be created for the different self-balancing approaches. This ranking system aggregates the ranks of each method across different parameters, with each method then ranked based on their respective total scores. In terms of the total number of epochs parameter, the constant weight approach aligns closely with the ratio approach and the original variance of SoftAdapt, while requiring fewer epochs compared to the normalized and weighted versions of SoftAdapt. The latter of these two appears to demand fewer epochs.

For the mean iteration time parameter, no significant difference is evident among the SoftAdapt methods or between the constant weight and ratio methods. Moreover, the SoftAdapt

group requires less time per epoch than the constant weight and ratio group. The same trend is observed concerning the standard deviation of the time per epoch. For the absolute difference between the CG and atomistic simulated potential energy, all methods can be deemed equivalent. With respect to the coefficient of variance of the simulated temperature parameter, all methods, barring the normalized SoftAdapt, are analogous. They also all demonstrate a lower CV_T compared to the normalized SoftAdapt.

Lastly, for the MSD slope parameters, all methods, with the exception of the constant weight method, can be considered equal. All these methods also have lower mean values compared to the constant weight method.

Given the above comparisons, the self-balancing methods can be ranked as follows:

1. Weighted SoftAdapt
2. Constant Weights
3. Ratio/Original SoftAdapt
4. Normalized SoftAdapt

The ranking of the self-balancing method is based on the ranking of each method on each specific parameter presented in Table 9.

Table 9: Ranking of each self-balancing method based on individual parameters.

Method	N_{ep}	$<t_{ep}>$	$\sigma_{t_{ep}}$	ΔU	$CV_{U_{CG}}$	$<T>$	CV_T	K-S D	K-S p-value	MSD slope	Total score
Constant Weights	1	1	4	2	2	1	1	3	1	3	19
Ratio	2	1	3	5	3	1	2	2	1	1	21
Original SoftAdapt	3	1	3	4	3	1	2	2	1	1	21
Weighted SoftAdapt	4	1	2	1	1	1	2	2	1	2	17
Normalized SoftAdapt	5	1	1	3	2	1	3	1	1	4	22

These findings highlight that, even without any optimization of the weighted version of SoftAdapt's hyperparameters, it can deliver superior results compared to the optimized constant weights approach.

Table 10: Results of the pairwise Tukey's test for the parameters that present p-value equal or below 0.35 in the Kruskal-Wallis test.

Group A	Group B	Total epochs			Iteration time			s_itertime			DU			CVT			MSD slope		
		meandiff	p-adj	reject	meandiff	p-adj	reject	meandiff	p-adj	reject	meandiff	p-adj	reject	meandiff	p-adj	reject	meandiff	p-adj	reject
constant	ratio	77.0	0.9874	False	-0.2504	0.9551	False	0.014	0.9992	False	7.2407	0.8313	False	16.2367	0.8732	False	-0.0184	0.7769	False
constant	norm SOA	797.0	0.0005	True	-12.9724	0.0	True	-0.2943	0.006	True	4.3809	0.9679	False	50.8303	0.0515	True	-0.0402	0.1266	True
constant	SOA	207.2	0.6824	False	-12.5234	0.0	True	-0.3805	0.0	True	1.6274	0.9993	False	-0.6384	1.0	False	-0.0187	0.7696	False
constant	weight SOA	401.0	0.1178	True	-12.9432	0.0	True	-0.2206	0.0097	True	-0.4135	1.0	False	15.7006	0.8859	False	-0.0114	0.9508	False
ratio	norm SOA	720.0	0.0015	True	-12.722	0.0	True	-0.3083	0.0003	True	-2.8598	0.9934	False	34.5936	0.2896	True	-0.0218	0.6569	False
ratio	SOA	130.2	0.9183	False	-12.273	0.0	True	-0.3946	0.0	True	-5.6132	0.9244	False	-16.8752	0.8572	False	-0.0002	1.0	False
ratio	weight SOA	324.0	0.2734	True	-12.6928	0.0	True	-0.2347	0.0057	True	-7.6132	0.8019	False	-0.5362	1.0	False	0.007	0.9917	False
norm SOA	SOA	-589.8	0.0097	True	0.449	0.7247	False	-0.0862	0.5932	False	-2.7534	0.9943	False	-51.468	0.0477	True	0.0216	0.665	False
norm SOA	weight SOA	-396.0	0.125	True	0.0292	1.0	False	0.0737	0.7204	False	-4.7943	0.9559	False	-35.1297	0.2759	True	0.0288	0.3998	False
SOA	weight SOA	193.8	0.7319	False	-0.4198	0.7699	False	0.1599	0.0855	True	-2.0409	0.9982	False	16.339	0.8707	False	0.0072	0.9907	False

5 Conclusions and Open Problems

This investigation set out to devise a robust, reproducible framework capable of automating the process of pre-screening undertrained models and quantifying the performance of more promising candidates, a fundamental step in the quest to advance deep learning technologies. This objective is tackled via the creation of an evaluation and ranking protocol that takes into account the anticipated behavior of a model grounded on physical insights, and subsequently gauges its performance employing statistical tests. The methodology put forth here offers an opportunity to rank proposed methods based on quantified parameters. It presents a powerful tool that can sieve out suboptimal models early in the process, hence, saving valuable computational resources and time. Concurrently, it quantifies the performance of promising models with a statistical rigour that adds a layer of reliability and objectivity to the selection process. In essence, the proposed framework enables more efficient and effective identification of potential deep learning models for applications in molecular dynamics and physical problems, thereby providing a significant contribution to the field and setting a new standard in the evaluation process of these technologies. This, in turn, guides future research by highlighting areas of improvement and challenges that remain in the quest for the most accurate and computationally efficient models.

he developed methodology was implemented to discern the most effective among multiple self-balancing methods for a MOO problem. This problem was contextualized within the domain of self-adapting the weights associated with the two components of the loss function for a GCNN model. This particular GCNN model was tasked with simulating a force field for predicting the configurations of a CG molecular system. The underlying model used for this problem was rooted in the SchNet architecture, a deep learning framework specifically designed for atomistic systems. The SchNet architecture is noted for its capacity to model quantum interactions in molecules, making it suitable for tasks involving molecular dynamics. The dataset employed for the training process comprised multiple frames drawn from atomistic simulations of benzene molecules. The problem at hand involved the self-adjustment of the weights of the dual components of the GCNN model's loss function. This self-balancing of weights aimed to optimize the trade-off between the different objectives in the multi-objective optimization problem. Using the developed method, the performance of various self-balancing methods was evaluated in this context, revealing the most optimal approach for this specific problem.

The evaluation method developed in this study was applied to various self-adaptation approaches based on the loss components' value or gradient. These encompassed the linear scalarization constant weight approach, the ratio approach, and three distinct variations of SoftAdapt: original, weighted, and normalized. Under the constant weight approach, the weights attributed to the two loss terms remained fixed throughout the entire model training process. These constant weights had been previously optimized by training the model on a dataset derived from the atomistic simulation at a temperature lower than that employed in the current study. A subsequent investigation into the transferability of the optimized weights and the model's other hyperparameters revealed that the optimized weights be effectively transferred. On the other hand, the ratio approach established the weights of the loss terms based on the values of the loss terms from the previous training epoch. An examination of the loss components' values as they evolved during the training process revealed a pattern in which each term would alternately dominate the loss function. Lastly, the SoftAdapt approaches utilized the change ratio of the loss components to calculate their respective weights, with the objective being to enhance all the loss terms at a uniform rate. In order to apply the developed evaluation method to compare these different approaches, multiple models were trained using each of the self-adaptive

methods. To ensure the robustness of the comparison, each training sequence was initiated with different seed values, thereby providing a diverse set of training scenarios for each approach.

For the rigorous evaluation of the models under consideration, a comprehensive set of performance metrics was established. The selection of these evaluation parameters was conducted with an eye towards capturing a holistic picture of each model's performance and predictive capabilities. The first three parameters pertain directly to the model training process. They are: (1) The total number of training epochs required by each model: This measures the efficiency of the training process, with fewer epochs indicating a more streamlined training sequence. (2) The mean duration of each epoch: This gives an indication of how long, on average, each round of training takes. A shorter mean epoch duration signals a more efficient model. (3) The standard deviation of the epoch duration: This evaluates the consistency of the training process. A lower standard deviation indicates a more consistent, and thus reliable, training process. The remaining evaluation parameters focus on the predictive performance of the model, specifically its ability to simulate a CG molecular system. These parameters are (4) The absolute difference between the mean CG and simulated potential energies: This measures how accurately the model predicts potential energy. A smaller difference implies a closer match to the true potential energy. (5) The CG simulated temperature: This evaluates the model's ability to accurately predict the system's temperature. (6) The coefficient of variation of the CG simulated potential energy and temperature: This assesses the variability of the model's predictions in terms of both potential energy and temperature. Lower coefficients suggest more stable and reliable predictions. (7) The statistic and p-value of the Kolmogorov-Smirnov test between the CG and atomistic simulated pair correlation function: This evaluates the similarity between the CG and atomistic simulations. A lower statistic and higher p-value indicate a greater similarity. (8) The slope of the last 30% of the MSD function: This captures how effectively the model predicts the movement of particles over time. In essence, an ideal model would demonstrate a lower number of epochs, shorter epoch durations, and a lower standard deviation of the epoch duration. It would also exhibit minimal differences between the mean CG and simulated potential energies, an accurate simulation of CG temperature, low coefficients of variation for the potential energy and temperature, a low statistic and high p-value in the Kolmogorov-Smirnov test, and a high value on the MSD slope.

The findings derived from the Kruskal-Wallis test reveal that the various self-balancing approaches diverge specifically in relation to five factors. These include the cumulative number of training epochs, the coefficient of variance of the simulated temperature, the average duration of each training epoch, the standard deviation of the duration of each epoch, as well as the slope of the MSD. These conclusions corroborate the insights obtained from the box plot visualizations of the parameters. The consistency between these two analyses strengthens the reliability of these findings, providing a solid basis for the comparison of the different self-balancing methods. By revealing areas of divergence among the methods, the Kruskal-Wallis test outcomes play an instrumental role in guiding the subsequent stages of analysis, including the application of Tukey's test for pairwise comparisons.

Subsequent to the Kruskal-Wallis test, a pairwise Tukey's test was implemented to conduct a comparative analysis between the distinct self-balancing approaches based on the selected parameters. Through this statistical method, it was possible to rank these different methods based on their overall performance. In accordance with the results from Tukey's test, the Weighted SoftAdapt method proved to be the most efficient approach. Following in the ranking were the Constant Weight approach and the Weighted SoftAdapt approach. They were classified as the second-best methods in terms of overall efficiency. The Ratio Approach was ranked third, together with the Original SoftAdapt whereas the Normalized SoftAdapt approach ranked at the

bottom. In light of these findings, it appears that the SoftAdapt method is a strong candidate for further investigation. Its ability to deliver superior performance without specific hyperparameter optimization suggests there might be more untapped potential for improvement within this method. With additional research and perhaps fine-tuning of its hyperparameters, the performance of the Original SoftAdapt method may be further optimized, potentially leading to even more efficient and effective model training.

Following the aforementioned results, several areas have been highlighted for further exploration. Firstly, the validation of these findings through an increased number of experiments emerges as a primary focus. By expanding the data set through additional trials, we can increase the robustness of the statistical analyses and strengthen the reliability of the conclusions drawn. This larger dataset would allow for a more detailed examination of trends and variances, thereby enriching the insights derived from our study. Secondly, refining the evaluation parameters warrants consideration. Existing parameters could be enhanced by integrating additional physical information, thereby encapsulating a more holistic perspective of the model's performance. Furthermore, assigning varying levels of significance to each parameter could introduce an element of weighting, which may reflect the real-world relevance or importance of each parameter more accurately. This augmentation would allow for a more nuanced understanding and could potentially improve the quality of conclusions drawn from the method, providing a more precise measure of the model's effectiveness. Lastly, an exploration into other self-adaptation approaches should be undertaken. Given the performance of the SoftAdapt methods in our study, further investigation into alternative approaches may reveal additional or even more effective strategies for optimizing the balance of the loss function.

In summary, the primary achievement of this research is the establishment of a novel and efficient framework to streamline the evaluation of various methodologies and approaches in the context of neural network models. This framework incorporates and leverages well-regarded statistical tools, effectively synthesizing empirical data and expert insights to draw robust, high-fidelity conclusions. By introducing a standard procedure for comparison, this method offers a significant reduction in the complexity and variability often associated with the evaluation process of different hyperparameters, methods, or approaches. It provides a uniform metric for comparison, thereby mitigating the inconsistencies arising from ad-hoc or disparate evaluation procedures. Furthermore, this approach not only refines the evaluation process but also fosters an environment conducive to knowledge-sharing and collaborative exploration. The method allows for the integration of expert insights in a structured and quantitative way, enabling a deeper understanding of the underlying processes and fostering a more nuanced interpretation of results. Through these advancements, the developed method represents a significant leap forward in our ability to systematically analyze, compare, and understand the performance of neural network models, a step that is crucial for their further refinement and application. Ultimately, the effectiveness of this approach lies in its ability to streamline the evaluation process, thereby promoting the more rapid development, refinement, and implementation of neural network models. Its application not only provides a more robust and standardized comparison but also guides future research directions and facilitates the process of knowledge synthesis and decision-making.

However, a few open problems persist. The possibility of improving the performance of the SoftAdapt method via hyperparameter fine-tuning remains unexplored. Moreover, the application of the evaluation and ranking methodology to other types of neural networks is yet to be tested. The development of an automated system that can perform this task with minimal human intervention also presents a challenge for future research.

References

[1] Michael P Allen and Dominic J Tildesley. *Computer simulation of liquids*. 2nd ed. Oxford, UK: Oxford University Press, 2017. DOI: 10.1093/oso/9780198803195.001.0001.

[2] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. 2nd ed. San Diego, CA: Academic Press, 2002. DOI: 10.1016/B978-0-12-267351-1.X5000-7.

[3] Niki Vergadou and Doros N. Theodorou. "Molecular modeling investigations of sorption and diffusion of small molecules in Glassy polymers". In: *Membranes* 9.8 (2019). ISSN: 20770375. DOI: 10.3390/membranes9080098.

[4] Martin Steinhauser and Stefan Hiermaier. "A Review of Computational Methods in Materials Science: Examples from Shock-Wave and Polymer Physics". In: *International Journal of Molecular Sciences* 10 (12 Dec. 2009), pp. 5135–5216. ISSN: 1422-0067. DOI: 10.3390/ijms10125135.

[5] Martin Karplus and Gregory A. Petsko. "Molecular dynamics simulations in biology". In: *Nature* 347 (6294 Oct. 1990), pp. 631–639. ISSN: 0028-0836. DOI: 10.1038/347631a0.

[6] Steve Plimpton. "Fast Parallel Algorithms for Short-Range Molecular Dynamics". In: *Journal of Computational Physics* 117.1 (1995), pp. 1–19. ISSN: 00219991. DOI: 10.1006/jcph.1995.1039. arXiv: 1712.04707.

[7] Albert P. Bartók et al. "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons". In: *Physical Review Letters* 104.13 (2010), pp. 1–4. ISSN: 00319007. DOI: 10.1103/PhysRevLett.104.136403. arXiv: 0910.1019.

[8] Kristof T. Schütt et al. "Quantum-chemical insights from deep tensor neural networks". In: *Nature Communications* 8 (2017), pp. 6–13. ISSN: 20411723. DOI: 10.1038/ncomms13890. arXiv: 1609.08259.

[9] Jörg Behler and Michele Parrinello. "Generalized neural-network representation of high-dimensional potential-energy surfaces". In: *Physical Review Letters* 98.14 (2007), pp. 1–4. ISSN: 00319007. DOI: 10.1103/PhysRevLett.98.146401.

[10] Stefan Chmiela et al. "Machine learning of accurate energy-conserving molecular force fields". In: *Science Advances* 3 (5 May 2017). ISSN: 2375-2548. DOI: 10.1126/sciadv.1603015.

[11] Linfeng Zhang et al. "Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics". In: *Physical Review Letters* 120.14 (2018). ISSN: 10797114. DOI: 10.1103/PhysRevLett.120.143001. arXiv: 1707.09571.

[12] Justin S. Smith et al. "Less is more: Sampling chemical space with active learning". In: *Journal of Chemical Physics* 148.24 (2018). ISSN: 00219606. DOI: 10.1063/1.5023802. eprint: 1801.09319.

[13] David Fooshee et al. "Deep learning for chemical reaction prediction". In: *Molecular Systems Design and Engineering* 3.3 (2018), pp. 442–452. ISSN: 20589689. DOI: 10.1039/c7me00107j.

[14] Peter Broecker et al. "Machine learning quantum phases of matter beyond the fermion sign problem". In: *Scientific Reports* 7.1 (2017), pp. 1–10. ISSN: 20452322. DOI: 10.1038/s41598-017-09098-0. arXiv: 1608.07848.

[15] Zhao Chen et al. “GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks”. In: *35th International Conference on Machine Learning, ICML 2018* 2 (2018), pp. 1240–1251. arXiv: 1711.02257.

[16] A. Ali Heydari, Craig A. Thompson, and Asif Mehmood. “SoftAdapt: Techniques for Adaptive Loss Weighting of Neural Networks with Multi-Part Loss Functions”. In: (2019). DOI: 10.48550/arXiv.1912.12355.

[17] Eleonora Ricci et al. “Developing Machine-Learned Potentials for Coarse-Grained Molecular Simulations: Challenges and Pitfalls”. In: ACM, Sept. 2022, pp. 1–6. ISBN: 9781450395977. DOI: 10.1145/3549737.3549793.

[18] Eleonora Ricci and Niki Vergadou. “Integrating Machine Learning in the Coarse-Grained Molecular Simulation of Polymers”. In: *The Journal of Physical Chemistry B* 127.11 (2023), pp. 2302–2322. ISSN: 1520-6106. DOI: 10.1021/acs.jpcb.2c06354.

[19] In Chul Yeh and Max L. Berkowitz. “Ewald summation for systems with slab geometry”. In: *Journal of Chemical Physics* 111.7 (1999), pp. 3155–3162. ISSN: 00219606. DOI: 10.1063/1.479595.

[20] Ilya Sutskever et al. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 2013, pp. 1139–1147.

[21] James Bergstra et al. “Hyperopt: A Python library for model selection and hyperparameter optimization”. In: *Computational Science and Discovery* 8.1 (2015). ISSN: 17494699. DOI: 10.1088/1749-4699/8/1/014008.

[22] Andrew R Leach. *Molecular modelling: principles and applications*. 2nd ed. Harlow, UK: Prentice Hall, 2001. DOI: 10.1021/jm970383k.

[23] Mark E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation. By Mark E. Tuckerman.* Oxford University Press, 2010. DOI: 10.1002/anie.201105752.

[24] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, Apr. 2004. ISBN: 9780521825689. DOI: 10.1017/CBO9780511816581.

[25] Mark James Abraham et al. “GROMACS: High-performance molecular simulations through multi-level parallelism from laptops to supercomputers”. In: *SoftwareX* 1-2 (Sept. 2015), pp. 19–25. ISSN: 23527110. DOI: 10.1016/j.softx.2015.06.001.

[26] Aidan P. Thompson et al. “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales”. In: *Computer Physics Communications* 271 (Feb. 2022), p. 108171. ISSN: 00104655. DOI: 10.1016/j.cpc.2021.108171.

[27] Romelia Salomon-Ferrer, David A. Case, and Ross C. Walker. “An overview of the Amber biomolecular simulation package”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 3 (2 Mar. 2013), pp. 198–210. ISSN: 17590876. DOI: 10.1002/wcms.1121.

[28] “CHARMM: The biomolecular simulation program”. In: *Journal of Computational Chemistry* 30 (10 July 2009), pp. 1545–1614. ISSN: 0192-8651. DOI: 10.1002/jcc.21287.

[29] “Scalable molecular dynamics on CPU and GPU architectures with NAMD”. In: *The Journal of Chemical Physics* 153 (4 July 2020), p. 044130. ISSN: 0021-9606. DOI: 10.1063/5.0014475.

[30] Erik Lindahl, Berk Hess, and David van der Spoel. “GROMACS 5.0: a high-throughput and highly parallel open source molecular simulation toolkit”. In: *Bioinformatics* 33.14 (2017), pp. 2020–2022. DOI: 10.1093/bioinformatics/btt055.

[31] Martin Karplus and J. Andrew McCammon. “Molecular dynamics simulations of biomolecules”. In: *Nature Structural Biology* 9 (9 Sept. 2002), pp. 646–652. ISSN: 10728368. DOI: 10.1038/nsb0902-646.

[32] R. Car and M. Parrinello. “Unified Approach for Molecular Dynamics and Density-Functional Theory”. In: *Physical Review Letters* 55 (22 Nov. 1985), pp. 2471–2474. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.55.2471.

[33] Siewert J. Marrink et al. “The MARTINI Force Field: Coarse Grained Model for Biomolecular Simulations”. In: *The Journal of Physical Chemistry B* 111 (27 July 2007), pp. 7812–7824. ISSN: 1520-6106. DOI: 10.1021/jp071097f.

[34] H. Fehske, R. Schneider, and A. Weiße, eds. *Computational Many-Particle Physics. Lecture Notes in Physics*. Vol. 739. Lecture Notes in Physics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-74686-7.

[35] Mehran Kardar. “Statistical Physics of Particles”. In: *Statistical Physics of Particles* (June 2007). DOI: 10.1017/CBO9780511815898.

[36] John D Chodera and Frank Noé. “Markov state models of biomolecular conformational dynamics”. In: *Current Opinion in Structural Biology* 25 (Apr. 2014), pp. 135–144. ISSN: 0959440X. DOI: 10.1016/j.sbi.2014.04.002.

[37] Ben Leimkuhler and Charles Matthews. *Molecular Dynamics*. Vol. 39. Springer International Publishing, 2015. ISBN: 978-3-319-16374-1. DOI: 10.1007/978-3-319-16375-8.

[38] Efrem Braun et al. “Best Practices for Foundations in Molecular Simulations [Article v1.0]”. In: *Living Journal of Computational Molecular Science* 1.1 (2019), pp. 1–28. DOI: 10.33011/livecoms.1.1.5957.

[39] H. J. C. Berendsen et al. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81 (8 Oct. 1984), pp. 3684–3690. ISSN: 0021-9606. DOI: 10.1063/1.448118.

[40] Hans C. Andersen. “Molecular dynamics simulations at constant pressure and/or temperature”. In: *The Journal of Chemical Physics* 72 (4 1980), pp. 2384–2393. ISSN: 00219606. DOI: 10.1063/1.439486.

[41] Shūichi Nosé. “A molecular dynamics method for simulations in the canonical ensemble”. In: *Molecular Physics* 52 (2 June 1984), pp. 255–268. ISSN: 0026-8976. DOI: 10.1080/00268978400101201.

[42] M. Parrinello and A. Rahman. “Crystal Structure and Pair Potentials: A Molecular-Dynamics Study”. In: *Physical Review Letters* 45 (14 1980), pp. 1196–1199. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.45.1196.

[43] Soumil Y. Joshi and Sanket A. Deshmukh. “A review of advancements in coarse-grained molecular dynamics simulations”. In: *Molecular Simulation* 47.10-11 (2021), pp. 786–803. DOI: 10.1080/08927022.2020.1828583.

[44] Youping Chen et al. “Assessment of atomistic coarse-graining methods”. In: *International Journal of Engineering Science* 49.12 (2011), pp. 1337–1349. DOI: 10.1016/j.ijengsci.2011.03.018.

[45] Ali Gooneie, Stephan Schuschnigg, and Clemens Holzer. “A review of multiscale computational methods in polymeric materials”. In: *Polymers* 9.1 (2017). ISSN: 20734360. DOI: 10.3390/polym9010016.

[46] B. M. Mognetti et al. “Coarse-grained models for fluids and their mixtures: Comparison of Monte Carlo studies of their phase behavior with perturbation theory and experiment”. In: *Journal of Chemical Physics* 130.4 (2009). ISSN: 00219606. DOI: 10.1063/1.3050353. arXiv: 0812.0331.

[47] Emiliano Brini et al. “Systematic coarse-graining methods for soft matter simulations-a review”. In: *Soft Matter* 9.7 (2013), pp. 2108–2119. DOI: 10.1039/c2sm27201f.

[48] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. Vol. 4. AMLBook New York, 2012.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[50] Garrett B Goh, Nathan O Hodas, and Abhinav Vishnu. “Deep learning for computational chemistry”. In: *Journal of computational chemistry* 38.16 (2017), pp. 1291–1307. DOI: 10.1002/jcc.24764.

[51] Junshui Ma et al. “Deep neural nets as a method for quantitative structure-activity relationships”. In: *Journal of chemical information and modeling* 55.2 (2015), pp. 263–274. DOI: 10.1021/ci500747n.

[52] Izhar Wallach, Michael Dzamba, and Abraham Heifets. “AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery”. In: *CoRR* abs/1510.02855 (2015). arXiv: 1510.02855.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[54] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[55] Michael I Jordan and Tom M Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260. DOI: 10.1126/science.aaa8415.

[56] Zhenqin Wu et al. “MoleculeNet: a benchmark for molecular machine learning”. In: *Chemical Science* 9.2 (2018), pp. 513–530. ISSN: 2041-6520. DOI: 10.1039/C7SC02664A. arXiv: 1703.00564.

[57] Xiang Fu et al. “Forces are not Enough: Benchmark and Critical Evaluation for Machine Learning Force Fields with Molecular Simulations”. In: (2022), pp. 1–25. arXiv: 2210.07237. URL: http://arxiv.org/abs/2210.07237.

[58] A San Solomon et al. “Neuron the memory unit of the brain”. In: *Journal of Computer Engineering* 17.4 (2015), pp. 48–61.

[59] Zaheer Allam. “Achieving Neuroplasticity in Artificial Neural Networks through Smart Cities”. In: *Smart Cities* 2.2 (2019), pp. 118–134. ISSN: 2624-6511. DOI: 10.3390/smartcities2020009.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386.

[61] Mohammad Mustafa Taye. “Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions”. In: *Computation* 11.3 (2023), p. 52. DOI: 10.3390/computation11030052.

[62] Nadia Jmour, Sehla Zayen, and Afef Abdelkrim. “Convolutional neural networks for image classification”. In: *2018 International Conference on Advanced Systems and Electric Technologies, IC$_A$SET*2018 (2018), pp. 397–402. DOI: 10.1109/ASET.2018.8379889.

[63] Neha Sharma, Vibhor Jain, and Anju Mishra. “An Analysis of Convolutional Neural Networks for Image Classification”. In: *Procedia Computer Science* 132 (2018), pp. 377–384. DOI: 10.1016/j.procs.2018.05.198.

[64] Farhana Sultana, Abu Sufian, and Paramartha Dutta. “Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey”. In: *Knowledge-Based Systems* 201-202 (2020), p. 106062. DOI: 10.1016/j.knosys.2020.106062. eprint: 2001.04074.

[65] Linwei Zhu et al. “Convolutional Neural Network-Based Synthesized View Quality Enhancement for 3D Video Coding”. In: *IEEE Transactions on Image Processing* 27.11 (2018), pp. 5365–5377. DOI: 10.1109/TIP.2018.2858022.

[66] Shiyang Liao et al. “CNN for situations understanding based on sentiment analysis of twitter data”. In: *Procedia Computer Science* 111 (2017), pp. 376–381. DOI: 10.1016/j.procs.2017.06.037.

[67] Jiajun Zhang and Chengqing Zong. “Deep Neural Networks in Machine Translation: An Overview”. In: *IEEE Intelligent Systems* 30.5 (2015), pp. 16–25. DOI: 10.1109/MIS.2015.69.

[68] Jie Zhou et al. “Graph neural networks: A review of methods and applications”. In: *AI Open* 1 (2020), pp. 57–81. DOI: 10.1016/j.aiopen.2021.01.001.

[69] Qiaoyu Tan, Ninghao Liu, and Xia Hu. “Deep Representation Learning for Social Network Analysis”. In: *Frontiers in Big Data* 2 (2019), pp. 1–10. DOI: 10.3389/fdata.2019.00002.

[70] Xiao Meng Zhang et al. “Graph Neural Networks and Their Current Applications in Bioinformatics”. In: *Frontiers in Genetics* 12 (2021), pp. 1–22. ISSN: 16648021. DOI: 10.3389/fgene.2021.690049.

[71] Patrick Reiser et al. “Graph neural networks for materials science and chemistry”. In: *Communications Materials* (2022), pp. 1–18. ISSN: 26624443. DOI: 10.1038/s43246-022-00315-6.

[72] Shiwen Wu et al. “Graph Neural Networks in Recommender Systems: A Survey”. In: *ACM Computing Surveys* 55.5 (2022). DOI: 10.1145/3535101. arXiv: 2011.02260.

[73] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[74] Petar Veličković et al. “Graph Attention Networks”. In: 2018. arXiv: 1710.10903 [stat.ML].

[75] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: 2018. arXiv: 1706.02216 [cs.SI].

[76] Rex Ying et al. “Graph convolutional neural networks for web-scale recommender systems”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2018, pp. 974–983. ISBN: 9781450355520. DOI: 10.1145/3219819.3219890. arXiv: 1806.01973.

[77] Guohao Li et al. “DeepGCNs: Can GCNs Go as Deep as CNNs?” In: *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob (2019), pp. 9266–9275. ISSN: 15505499. DOI: 10.1109/ICCV.2019.00936. arXiv: 1904.03751.

[78] Felix Wu et al. “Simplifying Graph Convolutional Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 6861–6871.

[79] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2020.2978386. arXiv: 1901.00596.

[80] David I. Shuman et al. “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”. In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98. ISSN: 10535888. DOI: 10.1109/MSP.2012.2235192. arXiv: 1211.0053.

[81] Zixue Xiang et al. “Self-adaptive loss balanced Physics-informed neural networks”. In: *Neurocomputing* 496 (2022), pp. 11–34. ISSN: 18728286. DOI: 10.1016/j.neucom.2022.05.015.

[82] Rafael Bischof and Michael Kraus. “Multi-Objective Loss Balancing for Physics-Informed Deep Learning”. In: *Neurocomputing* 496.2020 (2021), pp. 11–34. ISSN: 09252312. DOI: 10.13140/RG.2.2.20057.24169. arXiv: 2110.09813.

[83] D.F Jones, S.K Mirrazavi, and M Tamiz. “Multi-objective meta-heuristics: An overview of the current state-of-the-art”. In: *European Journal of Operational Research* 137.1 (2002), pp. 1–9. ISSN: 03772217. DOI: 10.1016/S0377-2217(01)00123-0.

[84] Ozan Sener and Vladlen Koltun. “Multi-task learning as multi-objective optimization”. In: *Advances in Neural Information Processing Systems* 2018-Decem.NeurIPS (2018), pp. 527–538. ISSN: 10495258. arXiv: 1810.04650.

[85] Michael Ruchte and Josif Grabocka. “Scalable Pareto Front Approximation for Deep Multi-Objective Learning”. In: *Proceedings - IEEE International Conference on Data Mining, ICDM* 2021-Decem (2021), pp. 1306–1311. ISSN: 15504786. DOI: 10.1109/ICDM51629.2021.00162. arXiv: 2103.13392.

[86] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks”. In: *Neural Networks* 3.5 (1990), pp. 551–560. ISSN: 08936080. DOI: 10.1016/0893-6080(90)90005-6.

[87] Sifan Wang, Yujun Teng, and Paris Perdikaris. “Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks”. In: *SIAM Journal on Scientific Computing* 43.5 (2021), A3055–A3081. ISSN: 1064-8275. DOI: 10.1137/20M1318043. arXiv: 2001.04536.

[88] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378.October (2019), pp. 686–707. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.10.045.

[89] Yoshua Bengio. “Practical Recommendations for Gradient-Based Training of Deep Architectures”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7700 LECTU. 2012, pp. 437–478. ISBN: 9783642352881. DOI: 10.1007/978-3-642-35289-8_26. arXiv: 1206.5533.

[90] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305. ISSN: 15324435.

[91] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Advances in Neural Information Processing Systems*. Vol. 4. 2012, pp. 2951–2959. ISBN: 9781627480031. arXiv: 1206.2944.

[92] Stefan Chmiela et al. “Towards exact molecular dynamics simulations with machine-learned force fields”. In: *Nature Communications* 9.1 (2018), p. 3887. ISSN: 2041-1723. DOI: 10.1038/s41467-018-06169-2. arXiv: 1802.09238.

[93] Eleonora Ricci et al. “Molecular Simulations and Mechanistic Analysis of the Effect of CO2Sorption on Thermodynamics, Structure, and Local Dynamics of Molten Atactic Polystyrene”. In: *Macromolecules* 53 (10 2020), pp. 3669–3689. ISSN: 15205835. DOI: 10.1021/acs.macromol.0c00323.

[94] K. T. Schütt et al. “SchNet: A continuous-filter convolutional neural network for modeling quantum interactions”. In: *Advances in Neural Information Processing Systems* 2017-Decem (1 2017), pp. 992–1002. ISSN: 10495258.

[95] K. T. Schütt et al. “SchNet - A deep learning architecture for molecules and materials”. In: *Journal of Chemical Physics* 148 (24 2018). ISSN: 00219606. DOI: 10.1063/1.5019779.

[96] Ask Hjorth Larsen et al. “The atomic simulation environment—a Python library for working with atoms”. In: *Journal of Physics: Condensed Matter* 29.27 (2017), p. 273002. DOI: 10.1088/1361-648X/aa680e.

[97] “Kolmogorov–Smirnov Test”. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 283–287. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_214.