UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Molecular dynamics of charged dipole particles using machine learning

Author:
Oleksandr KUKHAR

Supervisor:
PhD Taras PATSAHAN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Oleksandr KUKHAR, declare that this thesis titled, "Molecular dynamics of charged dipole particles using machine learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"God is a mathematician of a very high order, and He used advanced mathematics in constructing the universe."

Paul Dirac

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Molecular dynamics of charged dipole particles using machine learning

by Oleksandr KUKHAR

Abstract

The method of molecular dynamics finds applications in various areas such as pharmacology, polymer science, nanotechnology, chemical catalysis, and drug discovery. An efficient and fast prediction of positions and dynamics of particles is of great importance in order to reduce computational efforts. This thesis focuses on extending the existing SE(3)-transformer-based graph neural network (GNN) approach proposed by Fuchs et al.[1], which successfully employs a self-attention mechanism for point clouds to describe dynamics of charged particles. The extension developed in our study is aimed to improve an accuracy of molecular dynamics prediction for a more complex system consisting of particles with an orientation-dependent interaction and rotational degrees of freedom. As an example, a physical model presented as a fluid of charged particles bearing electric dipoles is examined. It is shown that our approach, which introduces a new attention mechanism, provides better accuracy in describing such systems compared to the original approach.

Acknowledgements

I thank God and all the people who are willing to risk their lives for Ukraine. Special thanks to my supervisor, Taras Patsahan.

Contents

List of Figures

List of Tables

List of Abbreviations

MD	**M**olecular **D**ynamics
BLEU	**B**i**l**ingual **E**valuation **U**nderstudy
CNN	**C**onvolutional **N**eural **N**etwork
GNN	**G**raph **N**eural **N**etwork
LJ	**L**ennard **J**ones
WCA	**W**eeks **C**handler **A**ndersen
MLP	**M**ulti **L**ayer **P**erceptron
MSE	**M**ean **S**quared **E**rror

Dedicated to Ukraine

Chapter 1

Introduction

Molecular dynamics (MD) is a powerful method of computer simulation used for the description of microscopic properties of various soft matter systems ranging from simple atomic and complex molecular liquids to systems of branched polymers and large macromolecules such as proteins. The MD method is based on the numerical solution of a set of Newton's equations of motion for interacting particles providing spatial coordinates and velocities of these particles depending on time. The obtained particle trajectories are used to calculate the physical properties of the systems under study. Model representation of a system usually requires consideration of a system with a large number of particles, and the period of time needed to obtain sufficiently accurate results can be very long.

Therefore, MD simulation can be very demanding in terms of computing resources and calculation time, which scales proportionally to the square of the number of particles. There are methods that allow one to significantly speed up such a calculation (e.g. nearest neighbors list algorithm) or/and one can parallelize the software code for its execution on multi-core CPU architectures, computing clusters, and GPUs. However, even this can be not enough, especially when it concerns systems with long-range interactions, such as systems of charged particles interacting via Coulomb interaction. Recently, MD acceleration methods using machine learning (ML) approaches have been considered. The combination of machine learning with MD simulations has shown promising results in accelerating the calculation of physical properties, especially for systems with long-range interactions and complex dynamics. Some algorithms combining ML with MD are already actively used [2], and some of them still need improvements. One of the promising ML approaches on the basis of graph neural network (GNN) combined with the self-attention mechanism for the points cloud is proposed in [1]. It was reported that this approach allows one to reach a high accuracy of prediction for the coordinates and velocities of charged particles with a relatively small training dataset.

In the present thesis, we develop code for complex particle simulating and generalize the approach described in [1] to the case of a system of particles that, in addition to charge, also bear electric dipole moments. For this, it is necessary to take into account, in addition to translational motion, also rotational motion, which includes the orientation of the particle's dipole moment vector and its rotational velocity. Since the studied system has additional degrees of freedom, we have to count that in order to improve the accuracy of the basic algorithm. So we consider an expansion of the set of weighting coefficients suggested in the original network. The proposed generalization will be tested on datasets generated by our code for MD simulation of charged dipolar particles in the rectangular box at different temperatures, similarly as it was done in [3]. Also, we consider the accuracy of predictions made by our approach depending on the temperature and the number of particles.

Chapter 2

Literature Overview and Related Works

2.1 Computer Simulation of Liquids [4]

Chapter 3, "Molecular Dynamics" holds paramount importance for our bachelor thesis, as it covers fundamental principles of molecular dynamics simulation, including the description of popular integrators for motion, force calculation, and dynamics of rotational movement. The accurate and effective implementation of an algorithm for dipole interaction is a crucial component of our thesis.

In order to model the behavior of a system of N particles interacting via a potential energy function V, the equations of motion can be expressed in terms of the Lagrangian equation of motion. This equation relates the generalized coordinates q and their time derivatives q' to the forces acting on the particles:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q_i'}\right) - \frac{\partial L}{\partial q_i} = 0$$

where L is the Lagrangian function defined in terms of the kinetic and potential energies of the particles, and i indexes the particles in the system. The generalized coordinates, q, can take different forms, depending on the nature of the system being modeled. For instance, Cartesian coordinates can be used for atoms, while internal coordinates can be utilized for molecules.

In order to solve the equations of motion, we need to apply an integration algorithm to calculate the positions and velocities of particles at each step. The velocity Verlet algorithm is the simplest and most widely used method for this purpose. It updates the position of each particle at each time step as follows:

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{1}{2}\mathbf{a}_i(t)\Delta t^2 + O(\Delta t^3)$$

where $\mathbf{r}_i$, $\mathbf{v}_i$, and $\mathbf{a}_i$ are the position, velocity, and acceleration of the i-th particle, respectively, at time t, and Δt is the time step. The velocity of each particle at time t + Δt is then calculated using the positions at times t and t + Δt as follows:

$$\mathbf{v}_i(t + \Delta t) = \frac{\mathbf{r}_i(t + \Delta t) - \mathbf{r}_i(t)}{\Delta t} + O(\Delta t^2)$$

The accuracy of the integration algorithm is crucial for the overall accuracy of the simulation, and various other algorithms, such as the leapfrog, Runge-Kutta, and Gear algorithms, can also be used depending on the specific needs of the system being modeled.

The calculation of forces is another essential aspect of MD simulation, as the forces acting on the particles determine their motion over time. In general, the force acting on each particle is calculated as the negative gradient of the potential energy function, as given by the following equation:

$$\mathbf{F}_i = -\nabla_i V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$$

where $\mathbf{F}_i$ is the force acting on particle i, V is the potential energy function of the system, and $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ are the positions of all N particles in the system. The potential energy function can take various forms depending on the nature of the interactions between particles, such as Lennard-Jones, Coulombic, or dipole-dipole interactions. The calculation of forces can be computationally intensive, especially for large systems, and various techniques have been developed to optimize the force calculation process, such as the use of neighbor lists or cutoff radii to limit the number of interactions that need to be calculated.

In addition to the translation of particles, MD simulations also need to take into account the rotational motion of molecules or other non-spherical particles. The authors describe two approaches for treating rotational motion: the use of Euler angles or the use of quaternion representations. Euler angles are a set of three angles that describe the orientation of a rigid body relative to a fixed frame of reference, and can be used to define the rotational motion of molecules. However, the use of Euler angles can lead to numerical instability and singularities for certain types of rotations, such as rotations about the z-axis. Quaternion representations, on the other hand, are more efficient and numerically stable for describing rotational motion, and can be used to avoid these issues. Quaternion representations involve the use of four parameters to describe the orientation of a rigid body, and can be easily converted to and from rotation matrices, which are used to transform vectors in three-dimensional space.

The treatment of rotational motion also involves the calculation of angular velocities and torques. The angular velocity of a particle is defined as the rate of change of its orientation with respect to time, and can be calculated using the following equation:

$$\omega(t) = \mathbf{S}^{-1}(\mathbf{Q}(t))\mathbf{L}(t)$$

where $\mathbf{Q}(t)$ is the quaternion representing the orientation of the particle at time t, $\mathbf{S}^{-1}$ is the inverse of the matrix that converts between the quaternion and rotation matrix representations, and $\mathbf{L}(t)$ is the angular momentum of the particle at time t. The angular velocity at time $t + \Delta t$ is then calculated using the orientation at times t and $t + \Delta t$ as follows:

$$\omega(t + \Delta t) = \mathbf{S}^{-1}(\mathbf{Q}(t + \Delta t)), [\mathbf{Q}(t + \Delta t) - \mathbf{Q}(t)]/\Delta t + O(\Delta t)$$

The calculation of torques is also important for describing the rotational motion of particles. Torques are defined as the rate of change of angular momentum with respect to time, and can be calculated using the following equation:

$$\tau(t) = \frac{d\mathbf{L}(t)}{dt}$$

where $\mathbf{L}(t)$ is the angular momentum of the particle at time t. The torque acting on a particle can be calculated as the negative gradient of the potential energy function with respect to its orientation.

In summary, Chapter 3 of "Computer Simulation of Liquids" provides a comprehensive overview of the fundamental principles of MD simulation, including the integration of equations of motion, the calculation of forces, and the treatment of rotational motion. The accuracy of MD simulations depends heavily on the integration algorithm used, the efficiency of the force calculation process, and the treatment of rotational motion. The use of quaternion representations can provide more efficient and numerically stable calculations of rotational motion. The principles and techniques described in this chapter are essential for accurate and efficient simulations of dipole interactions and other complex systems.

2.2 Attention Is All You Need [5]

In this work, the authors investigate the advantages of attention neural networks compared to recurrent and convolutional networks. They demonstrate that attention networks offer superior translation quality, higher parallelism, and shorter training time. To evaluate their models, they conducted experiments on the WMT 2014 English-to-German translation task, achieving a BLEU score of 28.4, surpassing existing results by more than 2 BLEU. Remarkably, their model also achieved a new state-of-the-art BLEU score of 41.0 on the WMT 2014 English-to-French translation task. Furthermore, besides the improved accuracy, the attention network demonstrated a significant reduction in training costs compared to previous state-of-the-art models in the literature.

They conducted the comparison between self-attention, recurrent, and convolutional layers for sequence transduction tasks, taking into account three criteria: computational complexity, parallelization, and path length between long-range dependencies. Results showed that a self-attention layer has a constant number of operations connecting all positions, while a recurrent layer requires $O(n)$ operations. It was found that self-attention is faster when the sequence length is smaller than the representation dimensionality, which is usually the case for state-of-the-art models. On the other hand, convolutional layers require a stack of $O(n/k)$ layers to connect all input and output positions, increasing the length of the longest paths. Additionally, self-attention could provide more interpretable models as attention distributions display behavior related to sentences' syntactic and semantic structure.

2.3 SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks [1]

The authors of this study conducted research that is similar with the objectives and focus of our thesis and a major part of our study is based on their work and code.
They used SE(3)-equivariant attention network to predict dynamics of charged particles. While their simulation code focuses solely on the Coulomb interaction between particles, our code takes into account also a core repulsion in the form of truncated and shifted repulsive part of Lennard-Jones force (known as WCA type of interaction). This inclusion ensures a more accurate and realistic representation of the intermolecular forces, making our code more corresponding with physical principles. The authors claim that the integration of attention networks with SE(3)-equivariance is highly effective. By leveraging self-attention, the SE(3)-Transformer model demonstrates high performance in handling large point clouds and graphs

with varying numbers of points saving reliability and consistency even when the input undergoes specific transformations.

2.4 Tensor field networks: Rotation- and translation- equivariant neural networks for 3D point clouds [6]

The paper introduces a novel family of neural networks called tensor field neural networks that exhibit richer equivariance to the symmetries of 3D Euclidean space. While convolutional neural networks are translation-equivariant and have contributed significantly to their widespread success, tensor field networks are locally equivariant to 3D rotations, translations, and permutations of points at every layer, making them more efficient than data augmentation to obtain 3D rotation-invariant output, and naturally encoding geometric tensors, which transform predictably under geometric transformations of rotation and translation.

The authors highlight three key advantages of equivariance: improved efficiency in achieving 3D rotation-invariant output, enhanced interpretability through consistent filter usage across different orientations and locations, and inherent encoding of geometric tensors for predictable transformation under rotation and translation.

The paper highlights three notable distinctions between tensor field networks and conventional CNNs. Firstly, tensor field networks utilize continuous convolutions on point clouds, considering both 3D coordinates and associated features. Secondly, the filters in these networks are determined by a combination of a learnable radial function and a spherical harmonic. Thirdly, the network's design is specifically developed to align with the algebraic properties of geometric tensors.

The authors show the versatility of tensor field networks in solving diverse tasks in geometry, physics, and chemistry. Originally developed for deep learning on atomic systems, these networks also offer potential for processing 3D images with rotation and translation equivariance. The paper presents a novel and promising approach to neural networks with wide-ranging applications across multiple domains.

The authors concentrate on a specific group of symmetry operations encompassing 3D space isometries and point permutations. They provide separate demonstrations of permutation, translation, and rotation equivariance, which collectively establish the network's equivariance to the different groups of transformations. The paper outlines the prerequisites for achieving permutation and translation equivariance, affirming that all introduced layers inherently possess these properties.

The authors introduce the group of 3D rotations, SO(3), parametrized by 3 numbers, and define its irreducible representations with dimensions $2l + 1$ for $l \in N$. They establish "rotation order" as the term for l, where orders $l = 0, 1, 2$ correspond to scalars, vectors, and symmetric traceless matrices. They achieve local rotation equivariance by employing specific convolution filters and decompose representations into irreducible representations for simplified analysis, utilizing Wigner D-matrices to map SO(3) elements to $(2l + 1)(2l + 1)$-dimensional matrices.

2.5 Learning Small Molecule Interaction[7]

In this study, the authors address a problem similar to ours but with a slightly different approach. Instead of predicting particle positions and velocities, they focus on predicting forces and energies using the SE(3)-Transformer model adapted from [1].

While it is possible to calculate energies and forces from predicted positions and velocities using formulas, directly predicting these values in molecular dynamics simulations offers several advantages. Firstly, it simplifies the overall process by eliminating the need for additional calculations or formulas, streamlining the workflow and reducing computational complexity. Secondly, direct prediction allows for the utilization of more sophisticated models, leading to improved accuracy and better capturing of the underlying physics. Furthermore, the direct prediction of energies and forces enables the model to learn complex patterns that may be difficult to capture using explicit formulas, resulting in more robust and reliable predictions. Also, some researchers propose an alternative approach where they predict only energies and then calculate the forces as the negative gradient of energies. The characteristics of the models may vary depending on the problem which has to be solved. In their study, the authors also explored various modifications of their initial model and found that deeper networks with additional SE(3)-Transformer layers exhibited the best performance.

Chapter 3

Methodology

3.1 Molecular dynamics and model system

Molecular dynamics is used to calculate positions (coordinates) and velocities of particles with time. For this purpose, the Newton's equations of motions are to be solved and the corresponding coordinates $\vec{r}_i(t)$ and velocities ($\vec{v}_i(t)$) for each of particles i can be calculated:

$$\vec{F}_i(\vec{r}_i) = m\frac{d^2\vec{r}_i}{dt^2},$$

where $\frac{d\vec{r}_i}{dt} = \vec{v}_i$ – the velocity of particle and $\frac{d\vec{v}_i}{dt} = \frac{\vec{F}_i(\vec{r}_i)}{m} = \vec{a}_i$ is the acceleration of particle. The force acting on particle i is calculated as the sum of all pair forces acting between particle i and other particles in the system:

$$\vec{F}_i(\vec{r}_i) = \sum_{i\neq j}^{N} \vec{F}_{ij}(\vec{r}_{ij}),$$

where $\vec{r}_{ij}$ is the vector-distance between particles i and j. In order to calculate forces the potential of pair interactions $U(\vec{r})$ are required:

$$\vec{F}_{ij}(\vec{r}_{ij}) = -\frac{dU(\vec{r}_{ij})}{d\vec{r}_i},$$

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ and $r_{ij} = \mid \vec{r}_{ij} \mid$. According to the third Newtons's law $\vec{F}_{ij}(\vec{r}_{ij}) = -\vec{F}_{ji}(\vec{r}_{ij})$.

In the case of orientation-dependent interaction, such as dipolar particles, pair potential is dependent not only on coordinates $\vec{r}_i$ and $\vec{r}_j$, but also on unit vectors of orientation of particles $\vec{u}_i$ and $\vec{u}_j$, i.e. $U(\vec{r_{ij}}, \vec{u}_i, \vec{u}_j)$. And instead of force, a torque has to be calculated:

$$\vec{T}_i = I\vec{\alpha}_i,$$

where I is moment of inertia and $\vec{\alpha}$ – angular acceleration.

A choice of the pair potentials of interaction determines a physical model of the system under study. In this thesis, a variation of the classical model of Stockmayer fluid [8] is considered one of the most popular theoretical models applicable to fluids with point electric dipoles. Besides the dipole-dipole interaction, we introduce the Coulomb interaction because of the particle charges (positive or negative). Finally, a soft repulsion at small distances equal to the size of particle core σ is taken into account via the Lennard-Jones (LJ) potential truncated and shifted at $r_{cut} = 2^{(1/6)}\sigma$ (also known as the WCA potential). A similar model of charged dipolar particles was studied recently in [9]. The total pair potential between two

particles (see Fig. 3.1), which includes all terms of interaction, is the following:

$$U(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j) = U_{WCA}\left(r_{ij}\right) + U_{cc}\left(r_{ij}\right) + U_{dc}\left(\vec{r}_{ij}, \vec{u}_i\right) + U_{dd}\left(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j\right),$$

where $U_{WCA}\left(r_{ij}\right)$ – the core-repulsive term (WCA potential), $U_{cc}\left(\vec{r}_{ij}\right)$ – charge-charge interaction (Coulomb potential), $U_{dc}\left(\vec{r}_{ij}, \vec{u}_i\right)$ – dipole-charged (or charge-dipole) interaction and $U_{dd}\left(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j\right)$ – dipole-dipole potential (Stockmayer potential). The detailed description of each of the terms of interaction, as well as the corresponding forces and torques are presented in the following Section **??**.

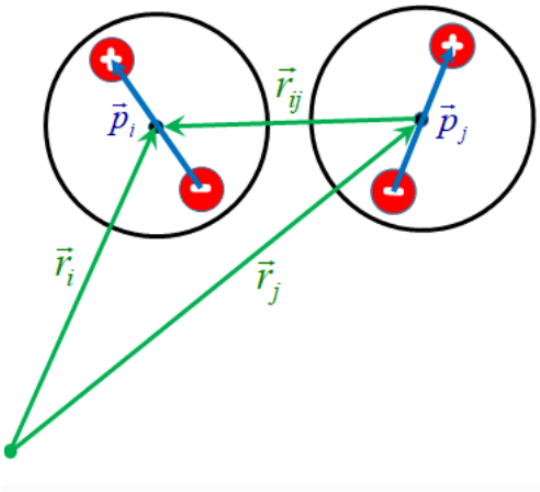

FIGURE 3.1: Model of dipolar charged particles ($|p_i|, |p_j| \ll |r_{ij}|$).

3.2 Model potential, forces and torques

3.2.1 WCA potential

The repulsive term of the pair potential of interaction is represented as repulsive part of the Lennard-Jones potential, which respectively cut and shifted as the following:

$$U_{WCA}\left(r_{ij}\right) = \begin{cases} 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] - 4\varepsilon\left[\left(\frac{\sigma}{r_{cut}}\right)^{12} - \left(\frac{\sigma}{r_{cut}}\right)^{6}\right], & r_{ij} \le r_{cut} \\ 0, & r_{ij} > r_{cut} \quad r_{cut} = 2^{1/6}\sigma \end{cases}$$

where σ is the diameter of particle core. This kind of potential is also known from the literature as the WCA pair potential.

The force corresponding to this term is:

$$\vec{F}_{ij}^{WCA}\left(\vec{r}_{ij}\right) = \begin{cases} \frac{24\varepsilon}{r_{ij}^2}\left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\vec{r}_{ij}, & r_{ij} \le r_{cut} \\ 0, & r_{ij} > r_{cut} \end{cases}$$

$$\vec{F}_{ij}^{WCA}\left(\vec{r}_{ij}\right) = -\vec{F}_{ji}^{WCA}\left(\vec{r}_{ij}\right)$$

3.2.2 Charge-Charge Interaction

The charge-charge interaction is described by the well-known Coulomb interaction:

$$U_{cc}\left(\vec{r}_{ij}\right) = \frac{q_i q_j}{r_{ij}}$$

Then the expression for force is

$$\vec{F}_{ji}^{cc}\left(\vec{r}_{ij}\right) = \frac{q_i q_j}{r_{ij}^3}\vec{r}_{ij},$$

$$F_{ji}^{cc}\left(r_{ij}\right) = -F_{ij}^{cc}\left(r_{ij}\right)$$

where q_i and q_j – the charges of particles i and j respectively. The charges can be negative or positive.

3.2.3 Charge-Dipole and Dipole-Charge Interaction

The charge-dipole interaction depends not only on the distance between particles, but also on the orientation of one of the particles having a dipole moment. This dipole moment is oriented along a unit vector $\vec{u}_i$ or $\vec{u}_j$.

$$U_{dc}\left(\vec{r_{ij}}, \vec{u}_i\right) = \frac{\mu_i q_j}{r_{ij}^3}\left(\vec{u}_i \cdot \vec{r_{ij}}\right) \quad U_{cd}\left(\vec{r_{ij}}, \vec{u}_i\right) = \frac{\mu_j q_i}{r_{ij}^3}\left(\vec{u}_j \cdot \vec{r_{ij}}\right)$$

where q_i and q_j are charges, μ_i and μ_j – dipole moment magnitudes, symbol "c" denotes charge and "d" denotes dipole.

The corresponding expressions for forces can be calculated as

$$\vec{F_{ij}^{dc}}\left(\vec{r_{ij}}, \vec{u}_i\right) = \frac{\mu_i q_j}{r_{ij}^3}\vec{u}_i - \frac{3\mu_i q_j}{r_{ij}^5}\left(\vec{u}_i \cdot \vec{r}_{ij}\right)\vec{r}_{ij} \quad F_{ij}^{cd}\left(r_{ij}\right) = -F_{ji}^{dc}\left(r_{ij}\right)$$

The torque for one of the particles (with dipole moment) is

$$\vec{T}_{ij}^{di}\left(\vec{r_{ij}}, \vec{u}_i\right) = \frac{\mu_i q_j}{r_{ij}^3}\left(\vec{u}_i \times \vec{r_{ij}}\right)$$

Vector of dipole moment in the model of a point dipole (see Fig. 3.1):

$$\vec{\mu}_i = z_i e \vec{p}_i = z_i e d_i \frac{\vec{p}_i}{|p_i|} = \mu_i \vec{u}_i$$

$\vec{p}_i$ - the distance between side charges in the original dipole.
d_i - the distance between charges in the original dipole.
μ_i - the scalar value of dipole moment.
$\vec{u}_i$ - unit vector of dipole (particle) orientation.

3.2.4 Dipole-Dipole Interaction

The dipole-diple interaction is used for modeling an interaction between two point dipoles. It is taken in the conventional form as

$$U_{dd}\left(\vec{r_{ij}}, \vec{u_i}, \vec{u_j}\right) = \frac{\mu_{ij}\vec{u_j}}{r_{ij}^3}\left[\left(\vec{u_i}\cdot\vec{u_j}\right) - \frac{3\left(\vec{u_i}\cdot\vec{r}_{ij}\right)\left(\vec{u_j}\cdot\vec{r}_{ij}\right)}{r_{ij}^2}\right]$$

where $\vec{u}_i$ and $\vec{u}_j$ are unit vectors of orientation of dipolar particles i and j respectively.

The forces $\vec{F}$ and and torques $\vec{T}$ for the dipole-dipole interaction can be obtained as following:

$$\vec{F_{ij}^{dd}}\left(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j\right) = \frac{3\mu_i\mu_j}{r_{ij}^5}\left(\vec{u}_i\cdot\vec{u}_j\right)\vec{r}_{ij} - \frac{15\mu_i\mu_j}{r_{ij}^7}\left(\vec{u}_i\cdot\vec{r}_{ij}\right)\left(\vec{u}_j\cdot\vec{r}_{ij}\right)\vec{r}_{ij} \qquad \vec{F}_{ij}^{dd}\left(\vec{r}_{ij}\right) = -\vec{F}_{ji}^{di}\left(\vec{r}_{ij}\right)$$

$$+\frac{3\mu_i\mu_j}{r_{ij}^5}\left[\left(\left(\vec{u}_j\cdot\vec{r}_{ij}\right)\vec{u}_i + \left(\vec{u}_i\cdot\vec{r}_{ij}\right)\vec{u}_j\right]$$

$$\vec{T}_{ij}^{dd}\left(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j\right) = -\frac{\mu_i\mu_j}{r_{ij}^3}\left(\vec{u}_i\times\vec{u}_j\right) + \frac{3\mu_i\mu_j\left(\vec{u}_j\cdot\vec{r}_{ij}\right)}{r_{ij}^5}\left(\vec{u}_i\times\vec{r}_{ij}\right)$$

$$\vec{T}_{ji}^{dd}\left(\vec{r}_{ij}, \vec{u}_i, \vec{u}_j\right) = -\frac{\mu_i\mu_j}{r_{ij}^3}\left(\vec{u}_j\times\vec{u}_i\right) + \frac{3\mu_i\mu_j\left(\vec{u}_i\cdot\vec{r}_{ij}\right)}{r_{ij}^5}\left(\vec{u}_j\times\vec{r}_{ij}\right)$$

3.2.5 Torque and rotational accelation

For the description of rotational motion one needs the expression for rotational acceleration:

$$\vec{\alpha}_i = \frac{1}{I}\left(\vec{G}_i - \vec{u}_i\left(\vec{u}_i\cdot\vec{G}_j\right)\right) - \vec{u}_i\cdot\vec{w}_i^2$$

$$\vec{G}_i = \sum_{i\neq j} g_{ij} = -\sum_{i\neq j}\frac{U_{dd}\left(r_{ij}, u_i, u_j\right)}{\mathrm{d}\vec{u}_i}$$

$$\vec{T}_i = \vec{u}_i\times\vec{G}_i = I\ddot{\vec{\phi}}_i$$

$$\dot{\vec{u}}_i = \dot{\vec{\phi}}_i\times\vec{u}_i$$

$$\dot{\vec{\phi}}_i = \vec{u}_i\times\dot{\vec{u}}_i$$

$$\left(\dot{\vec{\phi}}_i\right)^2 = w_i^2 = \left(\dot{\vec{u}}_i\right)^2$$

I - momentum of inertia, ϕ_i - angular velocity, α_i - angular acceleration

3.3 Integrator of Equations of Motion

For integration of equations of motion we used the velocity Verlet algorithm [4], because it is the simplest and most widely used method for this purpose. It updates the position, velocities, orientation and angular velocity of each particle at each time step as follows.
Translational motion:

$$\vec{r}_i(t+\delta t) = \vec{r}_i(t) + \vec{v}_i(t)\delta t + \frac{a_i(t)}{2}\delta t^2$$

$$\vec{v}_i(t+\delta t) = \vec{v}_i(t) + \frac{1}{2}\left(\vec{a}_i(t+\delta t) + \vec{a}_i(t)\right)\delta t$$

Rotational motion:

$$\vec{u}_i(t+\delta t) = \vec{u}_i(t) + \vec{w}_i(t)\delta t + \frac{1}{2}\vec{\alpha}_i(t)\delta t^2$$

$$\vec{w}_i\left(t+\frac{\delta t}{2}\right) = \vec{w}_i(t) + \frac{1}{2}\vec{\alpha}_i(t)\delta t$$

The rotational acceleration:

$$\vec{\alpha}_i(t+\delta t) = \frac{1}{I}\left[\vec{G}_i(t+\delta t) - \vec{u}_i(t+\delta t)\left(\vec{u}_i(t+\delta t)\cdot\vec{G}(t+\delta t)\right)\right] - \vec{u}_i(t+\delta t)\cdot\vec{w_i^2}\left(t+\frac{\delta t}{2}\right)$$

The angular velocity:

$$\vec{w}_i(t+\delta t) = \vec{w}_i\left(t+\frac{\delta t}{2}\right) + \frac{1}{2}\vec{\alpha}_i(t+\delta t)\delta t$$

3.4 Simulation

Computer simulations were performed in the cubic box at the constant volume $V = L^3$ and temperature T. The size of the box is set equal to $L = 8.0\sigma$ in all simulations. To control temperature the procedure of direct rescaling of particle velocities was applied every 100 simulation steps.

The boundary conditions are presented as repulsive (elastic) walls at the edges of the simulation box along all axis directions (no periodic boundary conditions). To describe collision of particles with the walls, a simple mirror reflection is used.

At the beginning of the simulation, we initialize the system by randomly assigning positions and orientations to the particles using a uniform distribution. Special care is taken to ensure that no overlap occurs between particles due to the strong repulsive force at distances less than particle diameter σ. Then, we uniformly distribute the velocities and angular velocities of the particles and rescale them during the simulation run to achieve a total system energy consistent with the desired temperature. Time step was set to 0.001 Employing the velocity Verlet integrator and utilizing the pair forces and torques described above, we calculate the total force and torque to determine the particle characteristics at each simulation time step. The

programming code was written by us in Python language on the basis of the code of Fuchs at al. [10]. In order to optimize the computation time of the simulation, we had to rewrite some functions of the numpy library, such as the cross product or dot product of vectors, because for a small number of particles, in our case, it is less than ten, direct multiplication functions work faster than similar functions from numpy which raise large functions for checks and optimization. This improvement allowed decreased time for simulation by 40 percent. The recorded system data includes particle positions, velocities, charge values, indications of wall reflections, angular velocities, and orientations. As the characteristics of consecutive frames are highly similar, we store information at regular intervals, in this case, every 100 time steps. Consequently, a simulation spanning 5000 steps results in only 50 entries in the output file.

3.4.1 Thermostat

Thermostating of the system is peformed using the velocity rescaling scheme. For this, the kinetic energies of translational and rotational motion have to be calculated: $K = K_t + K_r$

Translational motion: $K_t = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 = \frac{3}{2} N k_\mathrm{B} T_t$

Rotational: $K_r = \frac{1}{2}\sum_{i=1}^{N} I_i \omega_i^2 = N k_\mathrm{B} T_r$

Rescaling of translational and angular velocities:

$\beta_t = \left(\frac{3}{2}\frac{N k_\mathrm{B} T}{K_t}\right)^{\frac{1}{2}}$ $\quad \beta_r = \left(\frac{N k_\mathrm{B} T}{K_r}\right)^{\frac{1}{2}}$

$\beta_t v \to v$ $\quad \beta_r w \to w$

3.4.2 Reduced units

All quantities used in this thesis are presented in reduced units:

Mass:	$\frac{M}{m} \to m$	Time:	$t\sqrt{\frac{\varepsilon}{m\sigma^2}} \to t$	Charge:	$q\frac{1}{\sqrt{4\pi\,\grave{o}\,\sigma\varepsilon}} \to q$
Distances:	$\frac{\vec{r}}{\sigma} \to \vec{r}$	Velocity:	$\vec{v}\sqrt{\frac{m}{\varepsilon}} \to \vec{v}$	Dipole:	$\mu\frac{1}{\sqrt{4\pi\grave{o}\sigma^3\varepsilon}} \to \mu$
Density:	$\rho\sigma^3 = \frac{N}{V}\sigma^3 \to \rho$	Torque:	$\frac{\vec{T}}{\varepsilon} \to \vec{T}$	Force:	$\vec{F}\frac{\sigma}{\varepsilon} \to \vec{F}$

3.5 Benchmark

To validate the correctness and accuracy of the results proven by our code, we performed simulations using the widely-used LAMMPS molecular dynamics software. We utilized OVITO visualizations to provide a three-dimensional view of the system. Our comparisons were based on simulations involving four dipoles with temperature: $T = 0.1$ and timestep $\delta t = 0.001$ everywhere.

The trajectories of particles obtained by our code were compared with the analogous simulations performed using the LAMMPS package. While we observed some minor discrepancies in results after approximately $10,000$ steps, these differences may be attributed to variations in the algorithm of reflection from the walls. We can neglect this diverging considering that the original project only utilizes 5000 steps

by default, so our model works the best it can. The picture below presents a comparison between a LAMMPS simulation of four dipoles with 10,000 steps (left) and our simulation (right).

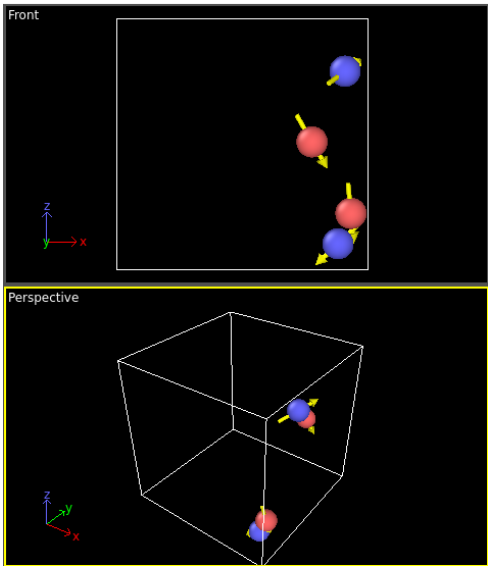

FIGURE 3.2: LAMMPS package.

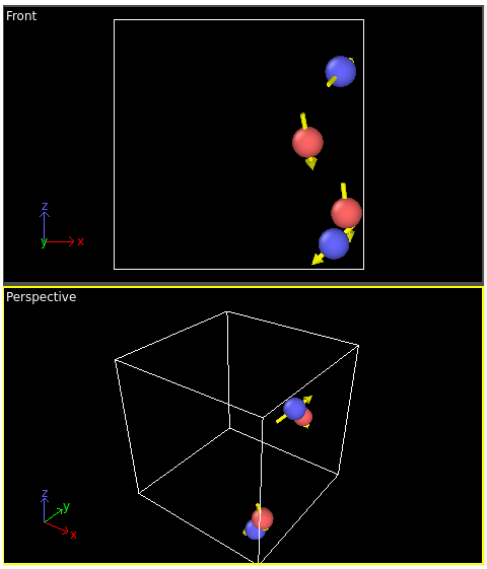

FIGURE 3.3: Our code.

3.6 Machine learning

3.6.1 General concept

The major goal of this work is to develop a machine learning approach capable of predicting molecular dynamics trajectories of interacting particles, on the basis of SE(3)-transformer concept and self-attention mechanism. The need for extension arises due to specific characteristics of the model examined in our study, wherein the particles interact through an orientation-dependent pair potential, i.e. dipole-dipole interaction. It has been observed that the original formulation of the SE(3)-Transformer based on Graph Neural Network (GNN) lacks accuracy in predicting the dynamics of such particles.

A system of N particles is presented as a complete graph with N nodes, each of them is characterized by such vector features as position, velocity, orientation and angular velocity, and scalar features – charge (negative/positive), dipole moment magnitude, collision with the box boundaries. The edges contain the relative distances between pairs of particles. We consider two cases: 1) charged particles *without* dipole moments; 2) charged particle *with* dipole moments. Using MD simulations described in Section 3.1-3.4 the corresponding datasets were generated to train the neural network for different number of particles and at different temperatures.

GNN is deep learning model that processes data in a graph structure. The convolutional operations performed on a graph are similar to those in a convolutional neural network. In graph neural networks, information passed between nodes similarly as in a convolutional neural network between adjacent pixels. Following message passing, the messages are aggregated to update the node data.

The SE(3)-Transformer architecture is designed to process 3D point clouds and incorporates the effective self-attention mechanism while maintaining equivariance constraints. These constraints guarantee that the predictions of network remain consistent under global roto-translational transformations of the input point cloud, leading to enhanced robustness and overall performance.

In this study, we apply and extend the SE(3)-Transformer architecture introduced by *Fuchs at al. 2020* [1]. In a conventional self-attention mechanism [5], each token is associated with three vectors: query vector $\mathbf{q}_i \in \mathbb{R}^p$, key vector $\mathbf{k}_i \in \mathbb{R}^p$ and value vector $\mathbf{v}_i \in \mathbb{R}^r$ for $i = 1, \ldots, N$, where low dimensional embeddings have dimensions r and p. These vectors are considered as outputs of learnable functions applied to the token feature vectors $\mathbf{f}_i \in \mathbb{R}^d$:

$$\mathbf{q}_i = h_q(\mathbf{f}_i), \qquad \mathbf{k}_i = h_k(\mathbf{f}_i), \qquad \mathbf{v}_i = h_v(\mathbf{f}_i),$$

On the basis of these vectors, the attention weights and attention-weighted value messages can be obtained:

$$Attn(\mathbf{q}_i, \{\mathbf{k}_j\}, \{\mathbf{v}_j\}) = \sum_{j=1}^{n} \alpha_{ij}\mathbf{v}_j, \qquad \alpha_{ij} = \frac{\exp(\mathbf{q}_j^T)}{\sum_{j'=1}^{n} \exp(\mathbf{q}_i^T k_{j'})}$$

The behavior of the function being learned remains unchanged or changes according to translational and rotational transformations of the inputs. This means that the function exhibits invariance or equivariance properties with respect to the SE(3) group of roto-translational transformations. Incorporating symmetry constraints explicitly into a neural network can be more efficient in terms of the number of learnable parameters and amount of data required for training, despite the fact that a general neural network can still learn to respect these symmetries. One example of a successful symmetry-aware architecture is the Tensor Field Network proposed by Thomas et al. [6] in 2018, which can map point clouds to point clouds in 3D while maintaining SE(3)-equivariance. The SE(3)-Transformer architecture has recently been extended to include the attention mechanism. In this enhanced architecture, the input consists of a feature vector field $\mathbf{f}$ that maps from $\mathbb{R}^3$ (three-dimensional space) to $\mathbb{R}^d$ (a space with dimension d). The feature vector field is defined on a discrete set of points in space, which can be referred to a as spatially distributed discrete finite point cloud:

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{f}_j \delta\left(\mathbf{x} - \mathbf{x}_j\right)$$

where δ is the Dirac delta-function, $\{\mathbf{x}_j\}$ are the 3D point coordinates, and $\mathbf{f}_j \in \mathbb{R}^d$ is concatenation of vectors $\mathbf{f}_j \in \mathbb{R}^{2l+1}$ of different degrees l of SO(3)-group irreducible presentations plus channels c:

$$\mathbf{f}_j = \bigoplus_{l \geq 0, c \in C_l} \mathbf{f}_j^{lc}$$

3.6.2 SE(3)-Transformer

The SE(3)-Transformer architecture comprises three main components: 1) edge-wise attention weights denoted as α_{ij}, which are SE(3)-invariant on each edge $\{ij\}$; 2) edge-wise SE(3)-equivariant value messages that propagate information between nodes; 3) a linear/attentive self-interaction layer.

In this study we extend a new type of attention mechanism by extending the original SE(3)-kernel attention, initially designed for working with 3D point clouds and graphs. This mechanism uses specialized attention weights to perform equivariant kernel conversions between the SO(3) vector types, allowing for the capture of complex angular relationships in the data.

$$\mathbf{W}_V^{lckd}(\mathbf{x}) = \sum_{J=|k-l|}^{k+l} \varphi_J^{lckd}(\|\mathbf{x}\|)\mathbf{W}_J^{lk}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right),$$

where $\varphi_J^{lckd}(\|\mathbf{x}\|)$ – completely unconstrained (learnable) radial functions and $\mathbf{W}_J^{lk}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$ is completely constrained angular basis kernels

$$\mathbf{W}_J^{lk}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right) = \sum_{m=-J}^{J} Y_{Jm}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)\mathbf{Q}_{Jm}^{lk},$$

$Y_{Jm}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$ – spherical harmonics and $\mathbf{Q}_{Jm}^{lk}$ – Clebsch-Gordan matrices.

By presenting the learnable weight kernel $\mathbf{W}_V^{lckd}(\mathbf{x})$ as a linear combination of non-learnable angular kernels $\mathbf{W}_J^{lk}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$ and scalar radial function coefficients $\varphi_J^{lckd}(\|\mathbf{x}\|)$ the transformation guarantees equivariance. This approach enables a valid conversion of k-degree vector to a l-degree vector:

$$\mathrm{f}_{\mathrm{out},i}^{lc} = \sum_{d\in C_l} w_V^{lcld}\mathbf{f}_{in,i}^{ld} + \sum_{k\geq 0, d\in C_k}^{L}\sum_{j\in\mathcal{N}_{i/i}} \alpha_{ijlckd}\mathbf{W}_V^{lckd}\left(\mathrm{x}_j - \mathbf{x}_i\right)\mathrm{f}_{in,j}^{kd},$$

where $\alpha_{ijlckd} = \alpha_{ij\omega}$ – conditioned attention weights

$$\alpha_{ij\omega} = \frac{\exp\left(\mathbf{q}_{i\omega}^{\mathrm{T}}\mathbf{k}_{ij\omega}\right)}{\sum_{j'\in\mathcal{N}_i/i}\exp(\mathbf{q}_{i\omega}^{\mathrm{T}}\mathbf{k}_{ij'\omega})}$$

$$\mathbf{q}_{i\omega} = \bigoplus_{l\geq 0, c\in\tilde{C}_l}^{\tilde{L}}\left(\sum_{d\in C_l} w_Q^{lcld\omega}\mathbf{f}_{in,i}^{ld}\right)$$

$$\mathbf{k}_{ij\omega} = \bigoplus_{l\geq 0, c\in\tilde{C}_l}^{\tilde{L}}\left(\sum_{k\geq 0, d\in C_k}^{L}\mathbf{W}_K^{lckd\omega}\left(\mathbf{x}_j - \mathbf{x}_i\right)\mathbf{f}_{in,j}^{kd}\right)$$

$$\mathbf{W}_K^{lckd\omega}(\mathbf{x}) = \sum_{J=|k-l|}^{k+l}\varphi_J^{lckd\omega}(\|\mathbf{x}\|)\mathbf{W}_J^{lk}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$$

We use the linear attentive self-interaction, which combines the self-interaction and non-linearity by replacing the learned scalars weights $w_{V,i}^{lcld} = w_V^{lcld}$ with attention weights obtained from a multi-layer perceptron (MLP):

$$w_V^{lcld} \text{ or } w_{V,i}^{lcld} = \mathrm{MLP}\left(\bigoplus_{l'\geq 0, c'\in C_{l'}, d'\in C_l'}^{L}\mathbf{f}_{in,i}^{l'c'T}\mathbf{f}_{in,i}^{l'd'}\right)$$

The SE(3)-invariance of these weights is attributed to the invariance of inner products between features, which undergo the same transformation within the given representation.

Chapter 4

Training and Results

4.1 Dataset

For models training we generated 18 datasets, 9 for dipoles and 9 for charges. Datasets differs by number of particles and temperature of simulation. We considered three different numbers of particles: 4, 6 and 8; and 3 different values of temperature 0.05, 0.1 and 0.25; so in result we got 9 datasets. Datasets for dipoles have lower number of simulation becasue their generating were much more time-consuming then for charges.
Dataset for charged dipoles training:

- Size of train dataset - 10000
- Size of test dataset - 2000
- Timesteps in simulation - 5000
- Timestep - 0.001
- Number of epochs - 250
- Number of particles - 4, 6, 8
- Temperature of the system - 0.05, 0.1, 0.25

Dataset for charges training:

- Size of train dataset - 15000
- Size of test dataset - 2500
- Timesteps in simulation - 5000
- Timestep - 0.001
- Number of epochs - 250
- Number of particles - 4, 6, 8
- Temperature of the system - 0.05, 0.1, 0.25

4.2 Metrics

As metrics we used MSE for each predicted characteristic, it is worth noting that our predicted variables as usual, are vectors:

$$MSE = \frac{\sum_{t=1}^{k} \frac{sum((predictedVariable_t - trueVariable_t)^2)}{d}}{k}$$

where d is the dimensionality of the variable and k is the number of timesteps for which the variable is predicted.

And for Test loss, we used averaged value of all losses:

$$TestLoss = \frac{\sum_{i=1}^{n} MSE_i}{n}$$

where n is the number of predicted variables.

4.3 Results

We compare results obtained using two schemes. First scheme is the previous version of SE(3)-Transformer proposed by *Fuchs et al.* [1], and we denote it as "single-alpha". The second one is the extended version of the SE(3)-Transformer proposed in the thesis, and it is denoted as "multi-alpha". To assess the performance of the model on our datasets, we conducted training using the original Fuchs model [1] on a dataset for charged particles. The model exhibited excellent performance in predicting the characteristics, confirming its suitability for more complex datasets and further experiments.

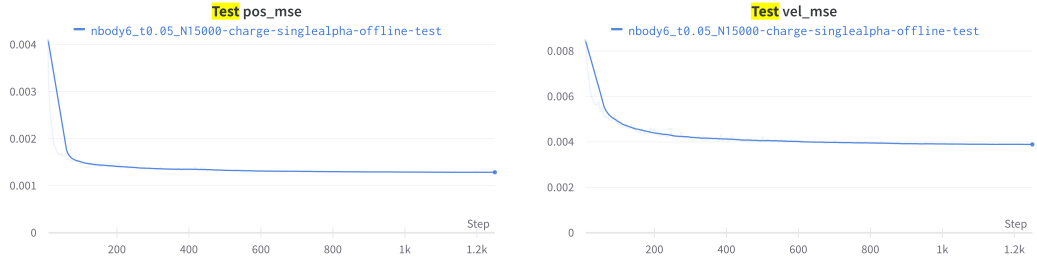

FIGURE 4.1: Prediction of position and velocities. MSE for one of the dataset of charged particles.

TABLE 4.1: Single-Alpha for charged dipoles

N\T	0.05	0.1	0.25	avg
4	0.0053	0.0108	0.0361	0.0174
6	0.0066	0.0141	0.0448	0.0218
8	0.0081	0.0157	0.0505	0.0248
avg	0.0066	0.0135	0.0438	0.0213

TABLE 4.2: Multi-Alpha for charged dipoles

N\T	0.05	0.1	0.25	avg
4	0.0044	0.0099	0.0309	0.0151
6	0.0055	0.0113	0.0341	0.0170
8	0.0063	0.0135	0.0431	0.0209
avg	0.0054	0.0116	0.0360	0.0177

In tables 4.1, 4.2, 4.3, 4.4 there are results of test loss of all experiments. Rows in the table represents a number of particles and columns represents a temperature.

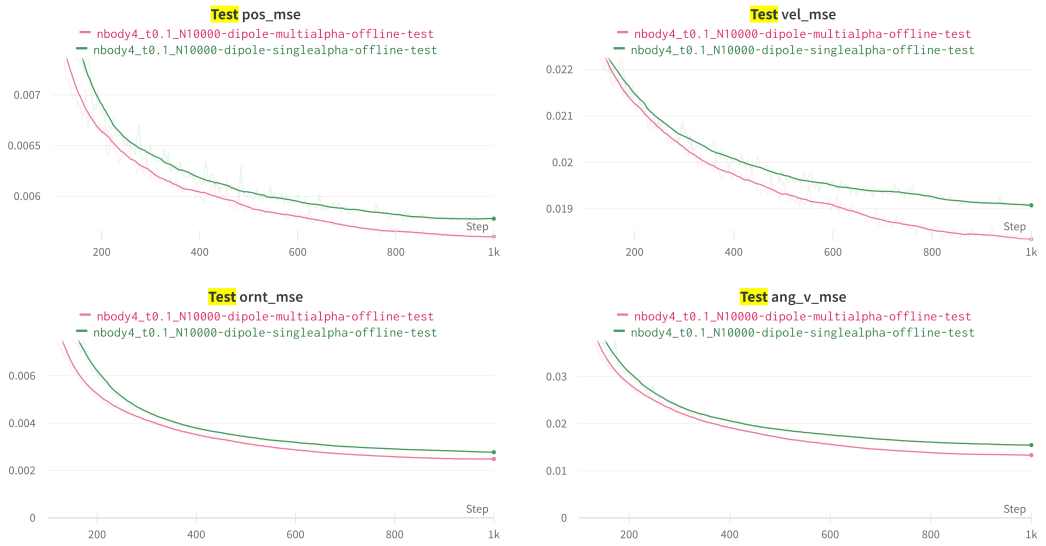

FIGURE 4.2: Prediction of positions, velocities, orientations and angular velocities. MSE for one of the charged dipoles dataset.

In the case of dipolar charged particles, one can see from the results that the accuracy worsens if to increase the temperature. This is related to an increase in average particle velocity at higher temperatures, resulting in larger translation distances and more frequent collisions occurring. Additionally, the test reveals that as the number of particles increases, there is a slight decrease in accuracy, which was somewhat surprising, because one could expect stronger correlation between test loss and number of particles.
The primary finding of the study indicates that the multi-alpha model yields significantly better results compared to the single-alpha model. On average, the multi-alpha model outperformed the single-alpha model by 17 percent.

TABLE 4.3: Single-Alpha for charges

N\t	**0.05**	**0.1**	**0.25**	**avg**
4	0.0036	0.0097	0.0363	0.0165
6	0.0027	0.0083	0.0309	0.0140
8	0.0026	0.0077	0.0316	0.0140
avg	0.0030	0.0086	0.0330	0.0148

TABLE 4.4: Multi-Alpha for charges

N\t	**0.05**	**0.1**	**0.25**	**avg**
4	0.0036	0.0099	0.0382	0.0172
6	0.0027	0.0084	0.0320	0.0144
8	0.0025	0.0076	0.0325	0.0142
avg	0.0030	0.0087	0.0342	0.0153

Regarding charged particles, the outcomes demonstrate a similar pattern. The accuracy did not show any significant changes as the number of particles increased, which may be due to the fact that the interaction of charges is much simpler than the interaction of charged dipoles.

On the Figure 4.3 the test losses of all experiments of training single-alpha and multi-alpha models are introduced.

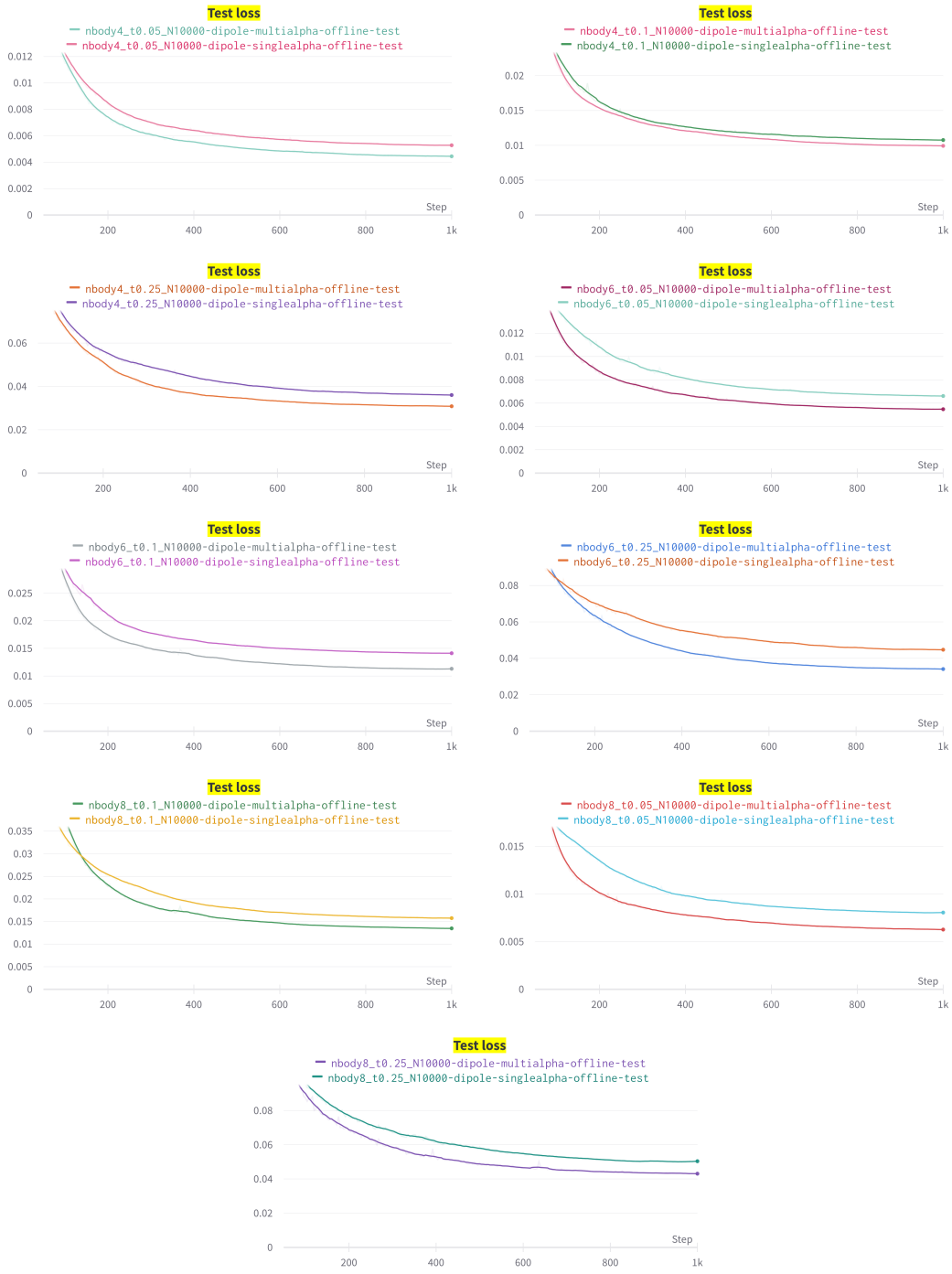

FIGURE 4.3: Charged dipoles results.

Chapter 5

Conclusion

In this study, we propose an extension of SE(3)-Transofmer based on Graph Neural Network and self-attention mechanism to accurately predict translational and rotational dynamics of the charged dipolar particles. The python code was adapted to perform molecular dynamics by taking into account orientation-dependent interaction and rotational type of motion. Molecular dynamics was performed to generate a dataset for the training of the neural network. To describe pair interaction between particles, we applied the conventional charge-charge, dipole-charge, dipole-dipole, and soft-core repulsive of the WCA-type terms of potential, and the corresponding forces and torques were calculated. The velocity Verlet algorithm was applied to integrate the trajectories of particles.

We validated the accuracy of our simulation by comparing it with established results. We generated diverse datasets with varying temperatures and particle numbers for dipoles and charged particles. Subsequently, we trained the SE(3)-equivariant attention model of two versions: single-alpha, following *Fuchs et al.* [1], and the one proposed in the thesis, which is a more complicated modification with the expansion of the set of weighting coefficients called multi-alpha. The results demonstrated a high level of accuracy for all predicted characteristics, with an average test loss of 0.0213 for single-alpha and 0.0177 for multi-alpha across all dipolar particle datasets. Notably, the multi-alpha model exhibited a significant improvement, achieving 17% percent better results. Moving forward, there are promising opportunities to enhance this model by training it on more complex datasets involving a larger number of particles and more intricate interaction types. The code of the project is available here.

Bibliography

[1] Fabian Fuchs et al. "SE (3)-Transformers: 3D Roto-Translation Equivariant Attention Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1970–1981. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/15231a7ce4ba789d13b722cc5c955834-Paper.pdf.

[2] Zhenwei Li et al. "Graph neural networks accelerated molecular dynamics". In: *The Journal of Chemical Physics* 156.14 (2022), p. 144103.

[3] Thomas Kipf et al. "Neural Relational Inference for Interacting Systems". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2688–2697. URL: https://arxiv.org/pdf/1802.04687.pdf.

[4] Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 2017. URL: https://levich.ccny.cuny.edu/koplik/molecular_simulation/AT2.pdf.

[5] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[6] Nathaniel Thomas et al. "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds". In: *arXiv preprint arXiv:1802.08219* (2018). URL: https://arxiv.org/pdf/1802.08219.pdf.

[7] Bryce Hedelius, Fabian B. Fuchs, and Dennis Della Corte. "Learning Small Molecule Energies and Interatomic Forces with an Equivariant Transformer on the ANI-1x Dataset". In: *arXiv preprint arXiv:2201.00802* (2022). URL: file:///home/oleksandr/Downloads/Learning_Small_Molecule_Energies_and_Interatomic_F.pdf.

[8] WH Stockmayer. "Second virial coefficients of polar gas mixtures". In: *The Journal of Chemical Physics* 9.12 (1941), pp. 863–870.

[9] Cameron J Shock et al. "Solvation Energy of Ions in a Stockmayer Fluid". In: *The Journal of Physical Chemistry B* 124.22 (2020), pp. 4598–4604.

[10] "Implementaion of SE(3)-Transformer". In: (). URL: https://github.com/FabianFuchsML/se3-transformer-public.