Analyzing protein-nanocluster interactions with graph-based machine learning for molecular dynamics

Master's Thesis, 17.5.2024

Author:

Anssi Sikoniemi

Supervisor:

Antti Pihlajamäki
Hannu Häkkinen

UNIVERSITY OF JYVÄSKYLÄ
DEPARTMENT OF PHYSICS

Abstract

Sikoniemi, Anssi
Master's thesis
Department of Physics, University of Jyväskylä, 2024, 52 pages.

In this work a custom graph convolutional network was succesfully constructed and trained to predict interaction energies in molecular dynamics simulations between $Au_{25}(SR)_{18}$ nanoclusters and BSA proteins based on their physical and chemical features. Data from molecular dynamics simulations was used as target data in supervised learning. The performance of this model was compared to a feed forward neural network with Weisfeiler-Lehman updates on graph form data. The energy terms predicted were the non-bonded Lennard-Jones and Coulombic terms for the force field used in the simulations. The models were created using the Keras Tensorflow package.

Both neural network architectures showed valid performance and the graph convolutional network based on localised spectral filters on graphs was at least as effective as the feed forward neural network with Weisfeiler-Lehman updates. The results show that these machine learning methods could be used in the future to improve molecular dynamics simulations by creating a better initialization for the simulations. To get more reliable results and generalise the models a larger data set would be required.

Keywords: nanoscience, machine learning, neural network, molecular dynamics, nanocluster, protein, force field, biophysics, graph convolutional network

Tiivistelmä

Sikoniemi, Anssi
Pro gradu -tutkielma
Fysiikan laitos, Jyväskylän yliopisto, 2024, 52 sivua

Tässä tutkielmassa tutkittiin vuorovaikutusenergioiden ennustamista $Au_{25}(SR)_{18}$ nanoklusterien ja BSA-proteiinien välillä kahdella eri neuroverkkoarkkitehtuurilla. Mallien kouluttaminen toteutettiin nanoklusterien ja proteiinien graafimuotoista esitystä hyödyntäen. Ennustetut vuorovaikutusenergiatermit olivat Lennard-Jones ja Coulombinen vuorovaikutusenergia simulaatioissa käytetylle voimakentälle. Ensimmäinen käytetty neuroverkkoarkkitehtuuri oli yksinkertainen eteenpäinsyöttävä malli, jossa datan esikäsittelyssä käytettiin Weisfeiler-Lehman -päivityksiä graafiesityksen parantamiseksi. Toinen käytetty koneoppimismalli oli graafikonvoluutioverkko, joka perustui graafien lokalisoituihin spektraalifilttereihin. Verkot rakennettiin hyödyntämällä Keras Tensorflow -pakettia.

Molempien mallien ennustuksien ja validaatiodatan välinen suhde oli hyvin lineaarinen. Molemmat mallit toimivat siis hyvin vuorovaikutusenergioiden ennustamiseen. Näiden tulosten pohjalta työssä käytettyä graafineuroverkkoa ja eteenpäinsyöttävää neuroverkkoa voisi hyödyntää molekyylidynamiikkasimulaatioiden alustamisen parantamiseen tulevaisuudessa. Suurin rajoittava tekijä tutkimuksessa oli käytetyn datan määrä. Luotettavampien tulosten saamiseksi ja mallien yleistämiseksi vaadittaisiin suurempi määrä dataa. Datamäärän lisääminen auttaisi luotettavampien johtopäätösten muodostamiseen myös siitä, kumpi neuroverkkoarkkitehtuuri on luotettavampi ja tehokkaampi vuorovaikutusenergioiden ennustamisessa.

Avainsanat: nanotiede, molekyylidynamiikkasimulaatiot, koneoppiminen, neuroverkko, nanoklusterit, proteiinit, voimakentät, biofysiikka, graafikonvoluutioverkko

Preface

First I would like to thank my main supervisor Antti Pihlajanmäki for his guidance and insightful ideas and comments especially when the course of the project needed adjustment. I also express my gratitude to Hannu Häkkinen for sending the email promoting the idea for this thesis and introducing me to this field. The work was done partly on the local computer cluster Oberon in Jyväskylä, so I also acknowledge the grants of computer capacity from the Finnish Grid and Cloud Infrastructure (persistent identifier urn:nbn:fi:research-infras-2016072533). A huge thank you also to all the friends that have been with me during these studies. And last but not least to my partner Eeva, I am grateful to you for always believing in me and for all the support you have given me during this project.

Jyväskylä May 17th 2024

Anssi Sikoniemi

Contents

Abstract **3**

Tiivistelmä **5**

Preface **7**

1 Introduction **11**

2 Theoretical background **15**

2.1 Molecular dynamics and force fields 15

2.2 Graph theory 19

2.3 Deep learning 20

2.3.1 Neural network concepts and the feed forward neural network 21

2.3.2 Graph convolutional networks 24

3 Data and neural network configuration **27**

3.1 Data and preprocessing 27

3.2 Neural network details and training 29

4 Results and model performance **33**

4.1 The FNN model with Weisfeiler-Lehman updates 33

4.2 The custom GCN model 37

4.3 Comparison of the models 41

5 Conclusions and outlook **43**

References **45**

1 Introduction

The role of machine learning (ML) methods in a variety of different fields has grown rapidly in recent times, especially when dealing with large scale data. Problems such as the modelling of protein-protein or protein-nanocluster interactions are usually studied with computational methods based on statistical physics such as molecular dynamics (MD) simulations. These methods lend themselves well for inclusion of machine learning methods. One reason for this is that even millisecond long MD simulations, which is a long timescale in the context of MD simulations, can generate terabytes of data [1]. There are several aspects of traditionally used computational methods that can be improved or replaced by ML methods. As an example one limitation of atomistic MD methods in the study of macromolecule interactions is that the interaction time between proteins can be in the order of minutes or even hours [2] while the MD methods are limited to a short time scale usually in the range of hunders of nanoseconds [3]. The idea of combining ML and MD methods has been around for decades now [4], but the recent interest in ML on the whole has naturally led to an increase in the use of these methods for the purposes of molecular simulations. In this work, I have employed and compared different machine learning architectures to predict the strength of interactions between proteins and nanoclusters based on data from MD simulations perfomed with GROMACS [5].

Many novel techniques for the analysis of protein-protein interactions have been developed in recent times [6–8], but protein-nanocluster interactions have not been studied to the same extent. The reason why protein-protein interactions have been modeled extensively compared to protein-nanocluster interactions is the fact that large amounts of experimental data for protein-protein complexes is widely available in the form of the protein data bank (PDB) [9]. To overcome the limited X-ray diffraction data for nanocluster-protein complexes there have been attempts to find unifying structural descriptors for the biological and bioinspired nanoscale complexes [10]. Understanding these nanoscale interactions is important in the formation of biomolecular complexes, a process which can be studied by molecular docking

methods where the problem is framed with the lock-and-key principle [11]. This method aims to predict the preferred orientation of molecules when they form a stable complex. The ligand molecule is thought as an key, which aims to open the lock, which is the binding domain of the other molecule, to form a stable complex. Molecular docking problems play an important role in drug design and in understanding biological processes. Molecular docking has multiple approaches, some of which have been combined with graph theoretical concepts [12] also used in this thesis. One of the problems in simulating these processes by solving the trajectories of the molecules from physical principles is that energy landscape is large and the total energy of the system has to be calculated after each move in phase space which leads to extended computation times. This is a problem that could be solved adding machine learning methods as a part of the molecular docking process. One of the goals of this thesis is to explore this possibility.

Machine learning methods can be used in the prediction of interaction parameters from the features of individual molecules or atoms which is also a goal of this thesis. Examples of this include the prediction of protein druggability [13] and the prediction of binding energies between hydrogen and nanoparticle catalysts which is essential in hydrogen production [14]. The data used in this thesis is from MD simulations of bovine serum albumin (BSA) proteins [15] and monolayer-protected gold nanoclusters (Au MPCs). BSA is an serum albumin protein derived from cows which often used as a model protein for other serum albumin proteins including the human one [16]. Monolayer-protected nanoclusters are a atomically precise metal nanoparticles which consist of an metallic core protected by an organic ligand layer with an interface structure between these two [17]. The clusters studied in this work are the thiolate protected $Au_{25}pMBA_{18}$ and $Au_{25}pMBSA_{18}$ clusters which are variants of the general thiolate protected $Au_{25}(SR)_{18}$ clusters. $Au_{25}(SR)_{18}$ nanoclusters have become the subject of wide interest because of their unique properties, stability, precise tunability and varius possible applications [18]. A detailed description of the nanoclusters studied in this work by Zhang et al. can be found in [19]. The $Au_{25}pMBA_{18}$ on the surface of the BSA protein is visualised in figure 1a. The chemical composition of the pMBA and pMBSA ligands are also depicted in figure 1b and 1c. The proteins and MPCs can be described by their physical and chemical features. One of the key points of this thesis is to use a graph representation with

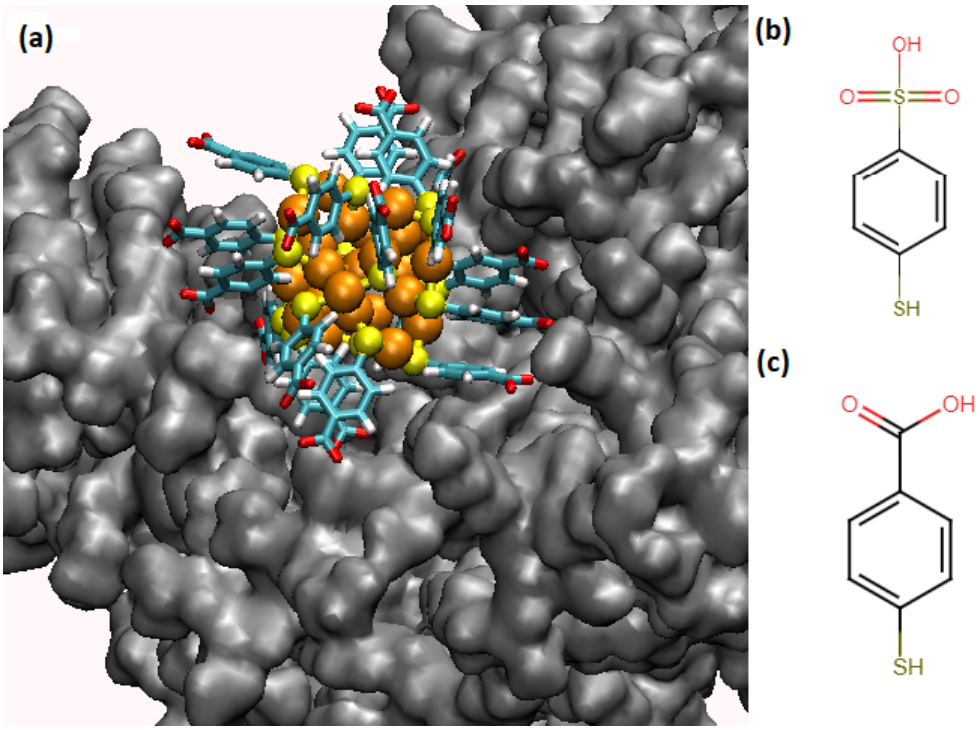


Figure 1. **(a)** Visualisation of the BSA protein (in grey) and the $Au_{25}pMBA_{18}$ nanocluster. Golden atoms represent gold, yellow ones represent sulphur, the blue parts are carbon-carbon bonds, white atoms represent hydrogen and red atoms are oxygen. **(b)** Chemical composition of the pMBSA ligand. **(c)** Chemical composition of the pMBA ligand.

embedded molecular features. There are many options for the graph representation of the proteins [20] but in this thesis the graph representation is formulated by using the alpha carbons of the residues as nodes. Each node contains features that represent the characteristics of the residue in question. The alpha carbons are then connected by an edge if they are within a cut-off distance from each other. This formulation has been used, for example, in the identification of backbone structures [21]. For the monolayer-protected gold nanoclusters, the nodes represent the functional head groups of the ligand molecules. When forming the paired representation for the protein-nanocluster complex, if the ligand binds to the residue an edge is formed from the ligand nodes to the residue node.

The machine learning methods used in this thesis are different neural network architectures. Neural networks have become the most common methods in ML in the last decade. Several architectures exist for the construction of neural networks

from which I have used on the most common one, the feed forward neural network (FNN) and a custom network designed to operate on graph-based data called graph convolution network (GCN) based on localized spectral filters on graphs [22]. The FNN offers an excellent comparison point to the GCN architecture because of its simplicity. In the FNN architecture graph form data is also used in the pre-processing of the data in the form of the Weisfeiler-Lehman scheme (WL) [23]. The WL-scheme is used propagate information from neighbouring nodes for a meaningful feature representation. Details of the WL-scheme are described section 2.2. The GCN on the other hand takes graph data straight as input and graph convolutions are used to extract useful representations of the graphs which is convenient when using the graph representation for the molecules. Graph-based machine learning has already been used to predict patterns in protein-protein interfaces [7], which inspired the development of the GCN model constructed in this thesis. Furthermore, the backbone of the custom GCN model was inspired by the work of Sai, Fu and Zhao on predicting binding energies and electronic properties of boron nitride fullerenes [24].

With all this in mind, the goal of this thesis is to see how different neural network architectures utilising graph data could be used to reliably estimate protein-nanocluster interactions based on MD data. The main use would be to create a better initialisation for the MD simulations to reduce computational cost. With a large and diverse enough dataset the models could also be used to predict binding affinities between proteins and nanocluster in general. I also provide a comparison between an FNN model with WL updates on the data and a GCN model with graph convolutions on the proteins graphs to see what is the optimal neural network architecture for this task. This comparison is also interesting from a computer science perspective as there has been discussion on the relationship of convolutions and the WL-scheme in graph processing [25].

2 Theoretical background

In this section I present the main theoretical concepts needed for understanding the work conducted in this thesis. First, a brief introduction to molecular dynamics simulations and force fields utilised in them is presented. This is vital as one of the goals is to see if the presented ML methods can be reliably used to predict interactions energies in MD simulations. Furthermore, an introduction to graph theory is provided which offers a base for understanding graph convolutions and the graph representation used for the proteins and nanoclusters. Lastly, an overview of the theory behind neural networks and the network architectures used in this work is presented.

2.1 Molecular dynamics and force fields

A molecular dynamic simulation consist of solving the classical equation of motion

$$m_i \frac{d^2 \boldsymbol{r}_i}{dt^2} = \boldsymbol{F}_i = -\frac{\partial}{\partial \boldsymbol{r}_i} U(\boldsymbol{r}_1, \boldsymbol{r}_2, ...\boldsymbol{r}_N), \tag{1}$$

for a system of N particles with potential energy $U(\boldsymbol{r}_1, \boldsymbol{r}_2, ...\boldsymbol{r}_N)$ dependent on the coordinates of the particles. m_i is the mass of an individual particle and $\boldsymbol{F_i}$ represents the total force acting on the particle. This system of N number of second order non-linear differential equations has to be solved numerically using a suitable integration algorithm. To do this you need a set of initial conditions and an expression for the interatomic potential. The initial conditions consist of the positions and velocities of the particles. For crystal structures the positions are known from crystallographic data and for disordered systems the positions can be set randomly or they can be set by melting an ordered configuration. The velocities can be initialised based on, for example, the Maxwell-Boltzmann distribution at a set temperature according to statistical mechanics. If a thermostat [26] is used to simulate a coupling of the system to an external heat bath initial velocities can also be set to zero. In the use of this method the system needs to be thermalised before doing the actual simulations

and making measurements. This means letting the simulations evolve for a certain amount of steps while the thermostat adjusts the velocities of the particles to reach a desired average temperature.

One aspect to also consider is the thermodynamical ensemble. Ideally the integration of equation (1) with these conditions will provide a trajectory for the particles in the system in a microcanonical ensemble. This means that the volume of the simulation cell V, the number of particles N and the total energy of the system E is kept constant. For this reason the microcanonical ensemble is sometimes referred to as the NVE ensemble. However errors introduced by the integration algorithm and distance cut-offs imposed on the forces can cause a slow drift in the systems total energy. Ultimately one also wants to compare the results from molecular dynamics to experimental results performed at a certain temperature or pressure. In the microcanonical ensemble the temperature of the system is not precisely known and thus it is better to work in a canonical ensemble. There are different ways of achieving this using thermostats [26], some with better physical justification than others. A common method is to use the Langevin thermostat [27]. The Langevin thermostat modifies the equation of motion by adding a frictional force term and a stochastic random force term according to the Langevin equation. This means that essentially the simulated particles can be considered to be moving in a sea of smaller particles with interactions between them. Another method which simulates a weak coupling of the system to a external heat bath is the Berendsen thermostat [28]. In the infinitely weak coupling limit this method returns the NVE ensemble.

Ideally the interatomic forces are derived from first principles by solving the electronic structure of the system using for example density functional (DFT) methods. However for large systems or for systems that have events that are outside of the timescale of these methods, including the protein-nanocluster interactions studied in this thesis, a more high level approximation called *force fields* are used. Force fields are mathematical expressions with an analytical form for the interatomic potential energy and a set of parameters entering into this form. These expressions are of course approximations of the true quantum mechanical potentials. Despite of this, an ideal force field offers a way to model the system in a way that is fast to compute and is sufficiently detailed to reproduce the properties of the studied system. Several

different force fields have been developed varying in complexity and specialised with different systems in mind. From these the GROMACS package includes CHARMM [29], AMBER [30], GROMOS [31] and OPLS [32] based force fields. A common expression for a force field used in molecular level simulations is in the form of

$$U = \sum_{bonds} \frac{1}{2} k_b (r - r_0)^2 + \sum_{angles} \frac{1}{2} k_a (\theta - \theta_0)^2 + \sum_{torsions} \frac{1}{2} V_n [1 + \cos n\phi - \delta] + \sum_{improper} U_{imp} + \sum_{\substack{non-bonded \\ i \neq j}} (4\epsilon_{ij}((\frac{\sigma_{ij}}{r_{ij}})^{12} - (\frac{\sigma_{ij}}{r_{ij}})^6) + \frac{q_i q_j}{r_{ij}}), \tag{2}$$

where the first four terms describe the intramolecular contributions to the potential energy (bond stretching, bending and dihedral and improper torsions) and the last term describes the intermolecular contributions. The intramolecular interactions are called the bonded interactions while the intermolecular interactions are called non-bonded interactions. In equation (2) k_b and k_a are force constant related to bond stretching and bending respectfully. Stretching and bending are modeled using harmonic functions where a larger force constant implies a more rigid bond. V_n is a force constant describing the energy barrier height associated with rotations around dihedral angles in a molecule. r_0 represents the equilibrium bond length i.e. the minimum of the potential energy curve with respect to the bond length for the harmonic oscillator representing the bond. θ_0 is the equilibrium value for the valence angle between atoms and θ is the non-equilibrium value. Relating to torsions, n is the dihedral multiplicity, ϕ is the dihedral angle, δ is the dihedral angle phase and U_{imp} is an correction term for torsions to take into account out-of-plane motions. All of these parameters have to be fitted to experimental data based on the specific system. The important fact to note from this is that force fields are empirical methods and their validation should be done by comparison to experimental data. Thus there is no one correct form for a force field but the functional form is chosen to get the most accurate result while keeping computational complexity at a reasonable level.

The first part of the intermolecular term known as the Lennard-Jones (LJ) equation models the Van der Waals interactions with attractive dispersion and repulsive Pauli exclusion terms. The $\frac{q_i q_j}{r_{ij}}$ term is the Coulomb interaction for atoms with assigned charges q_i and q_j. The 12-6 LJ potential is often used to represent the interatomic interactions with r_{ij} being the distance between two atoms, σ_{ij} being the distance at

which the LJ energy is at its minimum and the parameter ϵ_{ij} is the well depth. The LJ potential has a rich history in computational physics and chemistry. It has retained its position at the top of interatomic potentials despite its flaws ever since Fritz London argued that the dispersion energy decreases with r^{-6} under certain conditions based on the Schrödinger equation and Lennard-Jones adopted this finding to model interactions between molecules [33]. While the exponent -6 in the LJ potential has physical justification from quantum mechanics, the exponent -12 on the other hand is mainly used because it approximates the Pauli repulsion fairly well and is computationally efficient. This and the fact that the LJ potential is a pair potential, thus not taking into account many-body interactions, are the main disadvantages in its use.

The force field described here is also an additive force field. The meaning of this is that that the electrostatic energy of the system is the sum of all individual pair-wise Coulombic interactions due to the static partial atomic charges. This means that an mean-field approximation is performed and that the particles of interest are surrounded by an constant dielectric medium. In reality the electron density of an atom or a molecule is not static and this has to be taken into account in the development of new force fields [34]. Force fields that take this into account are called polarizable force fields. With the addition of additional terms or particles representing the electronic degrees of freedom, polarisable fields include electrostatic interactions with multi-body contributions. This allows the electronic structure of a molecule to change with respect to the local electric field.

In this work the part representing the non-bonded interactions in equation (2) is of importance as the aim is to predict the LJ and Coulombic contributions to the binding energy between the nanoclusters and the proteins with machine learning. The non-bonded interactions are the most computationally heavy part of solving the trajectories in MD simulations which offers motivation to improve the initial conditions for the simulations by ML methods so that less steps have to be calculated.

2.2 Graph theory

Graph theory is an active field of mathematics which has applications in a wide range of different fields from physics, computer science and economics to biology and social network analysis. In all of these fields, graph theory offers an easy to understand representation for complex networks. A graph, or more precisely an undirected graph, is a pair $G = (V, E)$, where V is a set of elements called *nodes* and E is an set of node pairs $\{v_1, v_2\}$ that are called *edges*. If the edges of the graph point only to one direction then the graph is called a directed graph but for the purposes of this thesis we will be working with undirected graphs.

There exists multiple ways of representing a graph. Usually for visualisation purposes graphs are represented as dots or circles connected by lines. Another way of representing graphs is the *adjacency matrix*. From the adjacency matrix we can tell which nodes are connected by edges. Formally for an undirected graph G with a node set $V = \{v_1, ..., v_n\}$ and edges E the adjacency matrix A is a square $n \times n$ matrix whose A_{ij} element is defined as

$$A_{ij} = \begin{cases} 1, \text{if } \{v_i, v_j\} \in E \\ 0, \text{otherwise.} \end{cases} \tag{3}$$

For simple graphs the diagonal elements are zeros because self-connections are not allowed, but for the spectral graph convolutions discussed later in this thesis we will add self-connections to our graph representations.

Another important concept in graph theory is the degree matrix. The degree matrix tells us the degree $\deg(v_i)$ of each node, that is the number of edges that terminate at each node. Thus as same as in the case of the adjacency matrix for a undirected graph G the degree matrix D is defined as

$$D_{ij} = \begin{cases} \deg(v_i), \text{if } i = j \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

In the context of this thesis the degree matrix tells us the number of connections for a certain residue α-carbon or ligand head. The degree matrix can also calculated

from the adjacency matrix as every row in the adjacency matrix corresponds to connections to a certain node. Thus $D_{ii} = \sum_j A_{ij}$. The adjacency matrix and degree matrix are needed for graph convolution described later.

In the context of the interactions between the thiolate protected nanoclusters and the BSA proteins the relevant interactions for forming a stable complex can be considered to be "local". Thus it makes sense to use a graph representation with information propagated from neighbouring nodes to the nodes associated with the relevant interactions. Machine learning methods also benefit from only using the most relevant information in the training data. One approach for data propagation in graphs is graph convolution introduced later in this work. Another approach is to use the continuous form of the *Weisfeiler-Lehman scheme* [23]. The original form of the Weisfeiler-Lehman scheme was intended as a test for graph isomorphism. For a graph G with continuous attributes for each vertex $v \in G$ and the initial attributes for each node denoted as $a^0(v)$ the WL-scheme is defined recursively as

$$a^{i+1}(v) = \frac{1}{2}\left(a^i(v) + \frac{1}{\deg(\mathrm{v})} \sum_{u \in \mathcal{N}(v)} w(v,u) a^i(u)\right), \tag{5}$$

where $\deg(v)$ is the degree of the node and $\mathcal{N}(v)$ represents the neighbouring nodes to node v. The edge weight are set to unity when not known $w(u,v) = 1$. The current WL iteration is denoted with i. This iterative process allows the propagation of information inside a graph by averaging over the neighbourhood of a node.

2.3 Deep learning

The subsection of machine learning called deep learning studies methods that utilize *neural networks*. Originally the study of neural networks was inspired by attempts to mathematically formalize the operation of neurons in the brain [35]. In the last decade, neural networks have become one of the most popular research topics in machine learning both in terms of their foundations and applications. This section aims to present the relevant concepts and neural network architectures relevant to this thesis. First I will discuss the most simple neural network architecture, the feed forward neural network, while discussing other important concepts in the area. Then I move on to the graph convolutional network that serves as a base for the custom

GCN model built for this thesis.

2.3.1 Neural network concepts and the feed forward neural network

The basic idea of a neural network is to approximate a function, which in an ideal case, maps input data x perfectly to some given output y, i.e. there exists a function f such that $y = f(x)$. This is achieved by stacking layers which consist of units called *neurons*. A neural network model always consist of at least an input layer, one hidden layer and an output layer. The dimension of the layer is defined by the number of neurons. Each layer in the model is connected by weighted transitions by which meaningful input values can be emphasized. In the context of graphs a neural network can be though of as an directed graph where the neurons act as nodes that has a set of input nodes on a set of output nodes with hidden layers of nodes in between. Nodes from the previous layer are then always connected to nodes of the next layer. The weighted transition between nodes are described by matrix multiplication. Along with this, a node specific bias is added to the result. With N_{in} number of input vectors H_i consisting of F number of features we can form an input matrix $H^l \in \mathbb{R}^{N_{in} \times F}$. Using this, the linear propagation rule from one layer to the next is of the form

$$H^{l+1} = W^l H^l + b^l, \tag{6}$$

where $H^{l+1} \in \mathbb{R}^{N \times f}$ is the output matrix with f features in the output vectors (dependent on the output size of the layer), $W^l \in \mathbb{R}^{f \times F}$ is a trainable weight matrix and b^l is a bias vector. The superscript l denotes that the matrices are layer specific. The problem with our model so far is that the output is always an linear combination of the inputs. This limits the output the propagation can produce. To perform more complex tasks than linear regression, we introduce a non-linear *activation function*. Denoting the activation function as $\sigma(\cdot)$, our layer-wise propagation rule in matrix form is expressed as

$$H^{l+1} = \sigma(W^l H^l + b^l) \tag{7}$$

Common activation functions used are the rectified linear unit ($ReLU$), the sigmoid function and the hyperbolic tangent (tanh) which are plotted in figure 2. The activation function should be chosen based on the nature of the task at hand (e.g. is it a classification or regression task or a regression task) and accordance to your dataset. Commonly sigmoid and tanh are used for classification tasks and ReLU

for regression tasks. Many other activation functions are available, some of which are altered version of the three mentioned here. All the activation functions have their own benefits and challenges. For example the sigmoid and tanh function are prone to the vanishing gradient problem while ReLU is non-differentiable and has problems with limited non-linearity. [36]

The next question is that how to set the values for weights and biases. This

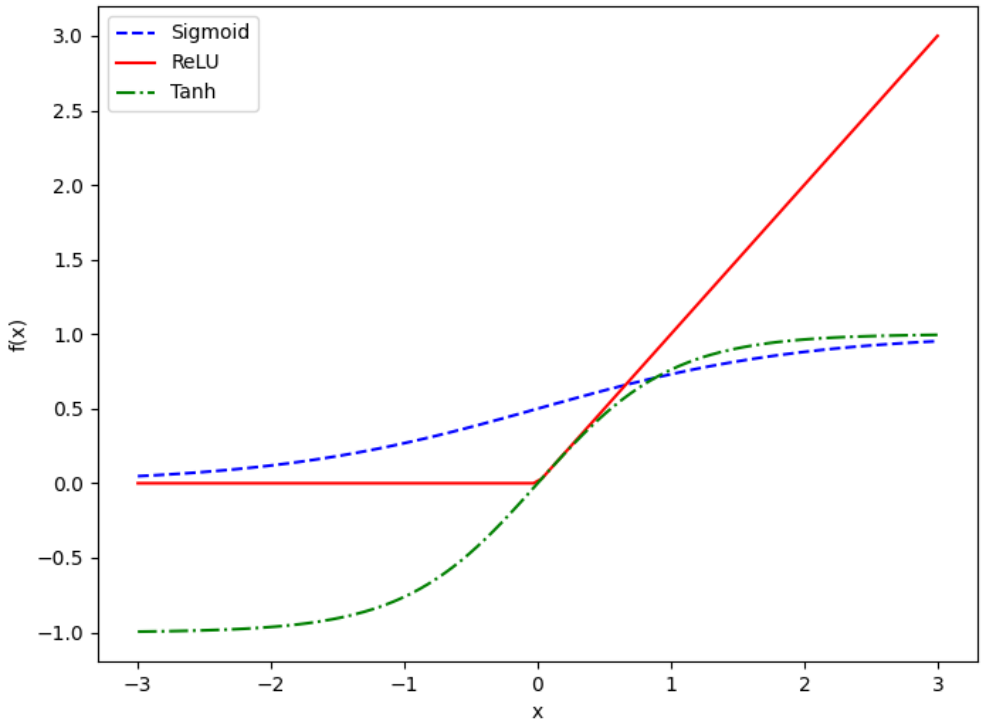


Figure 2. Plot of common activation functions ReLU, tanh and sigmoid.

is where we need to *train* the network. In this thesis training is done using supervised learning, meaning that for each feature set in the original input matrix X we have a corresponding set of target labels Y. The purpose of supervised learning is to approximate some function f that maps the input features to the labels $f(X) = Y$. We can now compare the output of the network the actual label. This is done by defining a *cost function* C which quantifies the difference between the output of the network and the actual labels. A multitude of different cost function exist, but common loss function for regression tasks is the mean absolute error (MAE) function,

which is defined as

$$C(W, b) = \frac{1}{N} \sum |y_i - y|, \tag{8}$$

where y_i is the predicted value for the i-th sample and y is the target value. This means that MAE calculates the average absolute distance between the predictions and the target values. As the output of our network is dependent on the weights W and biases b, the cost function is also dependent on these variables. In training the network, the aim is to find the weights and biases which make the cost function as small as possible. We can see that when the cost function is as small as possible the predictions are close to the target values. The adjustment of weights and biases to achieve this is done using a iterative gradient based algorithm. Usually this method is based on the backpropagation algorithm made famous by the 1986 paper by David Rumelhart, Geoffrey Hinton and Ronald Williams [37]. Backpropagation is based on the idea of defining an error δ_j^l for each neuron j in layer l to that neuron as

$$\delta_j^l = \frac{\partial C}{\partial H_j^l}, \tag{9}$$

where H_j^l is the input to that neuron. From this, using the chain rule, the equations that provide the gradient of the loss function with respect to the weights of the network can be derived. Then an optimizer, most commonly some form of the gradient descent algorithm, is used to adjust the weights based on the gradients. How much the weights are adjusted in each iteration is determined by the learning rate which is a tunable hyperparameter in the neural network. Multiple optimizers have been developed to address the problems related to gradient descent.

The optimisation of hyperparameters such as the learning rate is also a crucial part of the evaluation and construction of a machine learning model. Other than the learning rate hyperparameters usually consist atleast of the amount of layers in the model, the amount of neurons per layer, the batch size and the amount of epochs. The training data for the model is divided into different subsets to speed up training. The batch size tells the amount of data points which are processed before the model is updated. The number of epochs implies the amount of cycles that the training data passes trough the model completely. The number of epochs is an important hyperparameter in preventing over- or underfitting. Overfitting means that the model is trained too much on the training set and learns patterns specific

to that set. This prevents generalisation when the model is presented with unseen data. Underfitting means that the model is too simple and not trained enough so it can't make accurate predictions. To prevent underfitting and overfitting early stopping can be utilised. In early stopping the data is split in to test and training sets. After training with the training set predictions are made with the test set. The error related to the target values from the training set predictions and the test set predictions are then monitored. When the error from the test set stops decreasing training is stopped. If the validation error would be let to increase after this point, it would lead to overfitting.

The model that acts as a comparison to the graph convolutional network introduced in the next section the feed forward neural network (FNN). The FNN is the simplest and most common neural network architecture. It is formed by stacking layers which have a layer-wise propagation rule in the form of equation (7). In this thesis a fully connected feed forward network is used, meaning that every neuron in the previous layer is connected to every neuron in the next layer. The minimum amount of layers in a FNN is thus the input layer, one hidden layer and an output layer. It is important to note that in a FNN information flows only forwards i.e. only from the input layer towards the output layer.

2.3.2 Graph convolutional networks

Graph convolutional networks are a type of neural network that operates straight on graph form data and utilise convolutional operations on graphs. The word convolution may be familiar from image processing tasks where filters are applied to convolve the data. The principle in graph convolutions is the same but there are some major differences. One difference is that in traditional image convolutions, the convolution operation is defined by the windows size while graph convolution is defined solely by the edges of the graph. This means that the convolution operation is in practice defined by the properties of the graph itself. The mathematics of graph convolution become more involved than for image convolutions. The graph convolution network developed in this thesis is based on a network motivated by localized spectral filters

on graphs with a layer wise propagation rule

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}), \tag{10}$$

where $\tilde{A}$ is the adjacency matrix of the graph with added self-connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, W^l is the layer specific trainable weight matrix and $\sigma(\cdot)$ being the chosen activation function [22]. $H^l \in \mathbb{R}^{N\times f}$ is the output matrix with f features in the output vectors. This operation has the computational complexity of $\mathcal{O}(|E|Nf)$, where $|E|$ is the amount of edges in the graph. Thus large graphs can cause issues related to computational complexity. Spectral graph convolutions are described by the graph Fourier transform which can be considered to be an analogue of the regular Fourier transform. With graph convolution information can be propagated trough the graph-based on the structure of the graph. This takes in to account local and global features to form a meaningful representation for graph form data. Such as in the case of the FNN a GCN is also formed by stacking layers, but now with a propagation rule in the form of equation (10), and training the parameters based on a cost function.

3 Data and neural network configuration

The used dataset, the data loading process and the preprocessing steps performed on the data are described in the first part of this section. In the second part, the construction of the neural networks is described in detail. These details are important for understanding the practical differences between the model and the evaluation of the results presented later.

3.1 Data and preprocessing

The used dataset consisted of feature matrices for the protein residues and nanocluster ligands, graph structure information for the proteins in the case of the GCN model and data extracted from GROMACS MD simulations with values for the Lennard-Jones and Coulombic terms in the interaction energy from three simulations with 501 steps each. Each step is considered a datapoint when forming the arrays used as input to the neural networks with 1503 data points in total. There were 583 nodes representing the α-carbons in the protein residues and 18 nodes representing the ligand heads in the clusters. 39 protein features for each amino acid residue and 8 ligand features are included in total for each data point for the GCN model. The features for the residues include descriptors for its chemical composition (number of different atoms, molar mass, hydrophobicity etc.), the type of the residue and descriptors for the accessibility of the residue. The accessibility describes how open the residue is for interactions with other molecules. The features also include descriptors for the secondary structure of the protein. The secondary structure describes the local spatial conformation of the residue. It tells for example, how tightly the residue is bound to its local neighbourhood. The ligand features include the accessibility of the ligand and descriptors for the interactions with neighbouring ligands. For example the phenyl rings in the ligands can have pi-pi stacking. The protein features for the GCN model were not processed in any way but for the FNN model different amounts of WL-updates described by equation 5 were applied to the protein graphs to get the representation.

From the GROMACS MD data for the non-bonded energy terms for each simulation step were loaded in to act as validation data and target labels in supervised learning. In the case of the graph convolutional network adjacency matrices for the protein graphs were formed according to equation (3). The GROMACS data also contained information of how many pairs were formed in each step between the residues and the ligand heads. This was used to pre-process the data to remove steps that had no relevant interactions for training the network. Thus steps with no pairs were removed for both model architectures. Based on the pairing information "inter-adjacency" matrices were formed which represent the connections between the residues and the ligands. In graph formulation, if there was a pair between the ligand head and a residue then an edge was drawn between them and the inter-adjacency matrix was formed according to equation 3. For the FNN architecture the pairing information was used to form a paired representation of the ligand and residue features. This means that after the steps with no pairs were removed the feature arrays for the ligands and proteins were concatenated.

From the inter-adjacency matrix pair formation matrices for feature pairing were formed using the maximum number of pairs N_{pairs} across all simulation steps. They were formed separately for the ligands and proteins. Each row in the pair formation matrix corresponds to a pair in the simulations. Each element in the row corresponds to a ligand node in the ligand pair formation matrix and a residue node in the protein pair formation matrix. If there is a connection to a certain node in a pair, then the value for that element is one. Otherwise the elements are zeros. Thus the pair formation matrices are quite similar to the adjacency matrix. The motivation for the pair-formation matrices is better described in the next section where the GCN model is introduced in detail.

Because neural network training is based on a gradient descent algorithm the data also needs to be scaled to ensure that the gradient changes uniformly. If the data is not scaled then the different scales of the feature values would cause biases in training. Data scaling also makes the model usually converge faster. Min-max normalization [38] was used to scale the data. For the features data was scaled to the range of [0,1] and for the interaction energies to the range of $[-1,1]$. Thus the maximum value for

each feature was scaled to be 1 and the minimum value to be 0 and the rest of the values were scaled accordingly inbetween them. The scaling coefficients were saved to rescale the predictions back to the original scale.

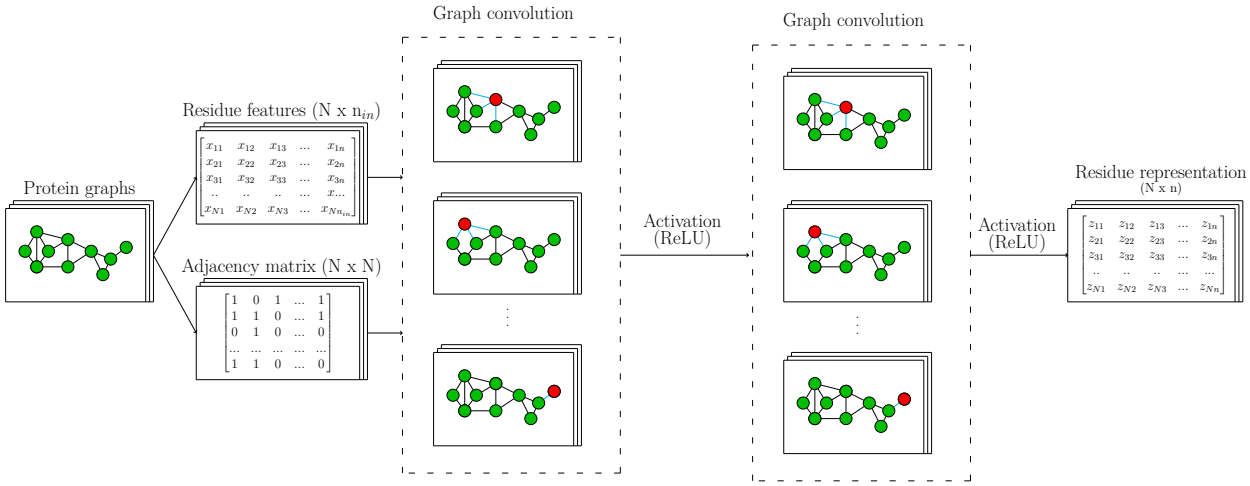
3.2 Neural network details and training

The neural networks were constructed using the Keras API for Tensorflow [39]. Keras has built in fully connected dense layers that make the construction of an simple FNN model convenient. The model with FNN architecture and Weisfeiler-Lehman updates was constructed by stacking these dense layers. Four dense layers were used with layer sizes (i.e. the amount of neurons per layer) 256, 128, 64 and 2. The first layer takes as input the paired features described in the earlier section. The interaction energies are used as target data. The output must be two dimensional because two energy terms are predicted and the output is summed along all pairs to get interaction energies for each step in the data. The FNN model takes in as a parameter the cut-off distance for the ligand head and residue interactions which was set to 10 Å. The amount of WL-updates on the data is a hyperparameter for the FNN model and several amount of them were tried as presented in the results.

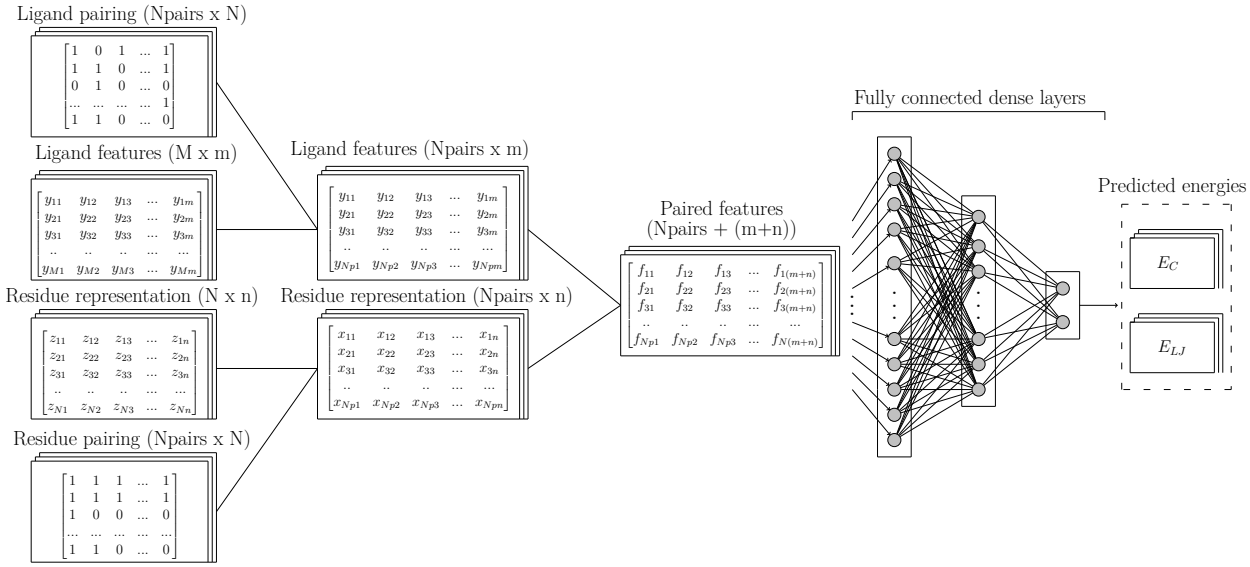
A schematic of the GCN model workflow is depicted in figure 3. It can be thought to consist of two parts. The first part in figure 3a is the graph convolutional part of the network that performs the graph convolutions using the adjacency and feature matrix provided as input. In the dimensions of the arrays N represents the amount of nodes in the protein graphs, n_{in} represents the original residue feature dimensionality and n is the feature dimension after convolutions. The convolution operations described by equation (10) were implemented using Tensoflows matrix multiplication for the convolution operation ($\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l$ part shown in equation (10)). A dense layer was added after this operation to apply weights and non-linearity to the convoluted graph. After this in the second part of the network as depicted in figure 3b the residue representation from the convolutional part is paired with the ligand features using pair-formation matrices. In the dimensions of the arrays M is the amount of residue nodes, m is the amount of ligand features and N_{pairs} is the maximum number of pairs in the simulations. The first reason for the use of the pair-formation matrices is that the features need to be fed into the dense layers in paired form. The motivation in the construction of the pair formation matrices is to make sure the dimensionality of

the paired features is correct. The lines between the different feature matrices and the pairing matrices in figure 3b represent matrix multiplications. After this feature matrices with the dimensions (N_{pairs}, m) and (N_{pairs}, n) are concatenated along the last dimension to form a paired representation of the features. The paired features are then fed into an fully connected dense layers. To make the comparison between the FNN and GCN models systematic the fully connected part at the end of the GCN model was kept the same size as the FNN network used with Weisfeiler-Lehman updates.

The activation function used for both models was ReLU and it was used for every layer except the output layer of the network because for regression tasks the constrainment induced by the activation function may limit the results. The cost function used in both cases was MAE described by equation (8) which is a common choice for regression tasks. For training and validation the dataset was split into a training set and a validation set. 5-fold cross validation was used in training the model. This means that the training data was split into five sets. Then each model was trained by using four of the sets as training data and one set as a test set. This resulted in five models trained on slightly different portions of the data. 106 data points were left aside as a validation set before forming the five sets for cross validation. Each of the models were evaluated using this validation set. Early stopping was utilised by plotting the mean validation errors from the cross validation sets and adjusting the amount of epochs to prevent under- or overfitting. The optimizer used for both models was Adam [40] and the learning rate used was 0.001.



(a)



(b)

Figure 3. a) Schematic depiction of graph convolutional layers in the custom GCN network. Adjacency matrices and feature matrices from each simulation step are taken as input and a new residue representation is acquired as output. **b)** Schematic depiction of feature pairing and FNN part of the custom GCN network. The residue representation from graph convolutional layers is paired with the ligand features and the paired features are fed into fully connected dense layers. Output consists of the non-bonded interaction energy terms E_C and E_{LJ}.

4 Results and model performance

In this section the results for the FNN model and the custom GCN model are first evaluated separately. In the case of the FNN, results are provided for models with different amounts of WL-updates applied on the data. For the GCN, results are provided for models with different amounts of convolutional layers and different sizes for them. Model performance and possible sources of errors are also discussed. At the end of the section, a comparison between the FNN and the GCN model is provided.

4.1 The FNN model with Weisfeiler-Lehman updates

Five models with the same feed forward neural network architecture were trained with each having a different amount of WL-updates on the data. The root-mean-square errors (RMSE) and Pearson coefficients (r) calculated between the energy values predicted by the model and the validation values from GROMACS simulations are presented in table 1. The values provided are the mean of the five predictions from training the model with different folds in cross-validation. The results are provided for the Lennard-Jones and Coulombic interaction energy term separately (E_C and E_{LJ}) as well as for the total energy (E_{tot}), which is the sum of the two terms. RMSE measures how much the predictions deviate from the validation values. The Pearson coefficient measures the linear correlation between the predictions and GROMACS values and for an ideal model that perfectly predicts the values the Pearson coefficient would be 1. The most accurate model is obtained with two WL-updates on the data as seen from the bolded values in table 1. The reason for this could be that additional WL-updates lead to a overly detailed description by capturing graph structures of properties specific to the training set and thus making it harder for the model to generalise. All of the models performed quite well, even though the Coulombic term especially has some variance in the RMSE, considering that the Pearson coefficient doesn't vary significantly between the models. The Coulombic term being harder to predict is an expected result based on the fact that the Coulombic interaction

Table 1. RMSE and Pearson coefficients for the mean FNN predictions

Number of WL updates	RMSE E_C (kJ/mol)	RMSE E_{LJ} (kJ/mol)	RMSE E_{tot} (kJ/mol)	Pearson r_C	Pearson r_{LJ}	Pearson r_{tot}
1	88.59	20.65	83.76	0.84	0.93	0.84
2	**66.82**	**19.27**	**67.15**	**0.86**	**0.93**	**0.85**
3	73.44	20.51	72.37	0.83	0.93	0.83
4	75.33	22.52	73.53	0.83	0.93	0.82
5	73.29	19.82	72.20	0.84	0.93	0.84

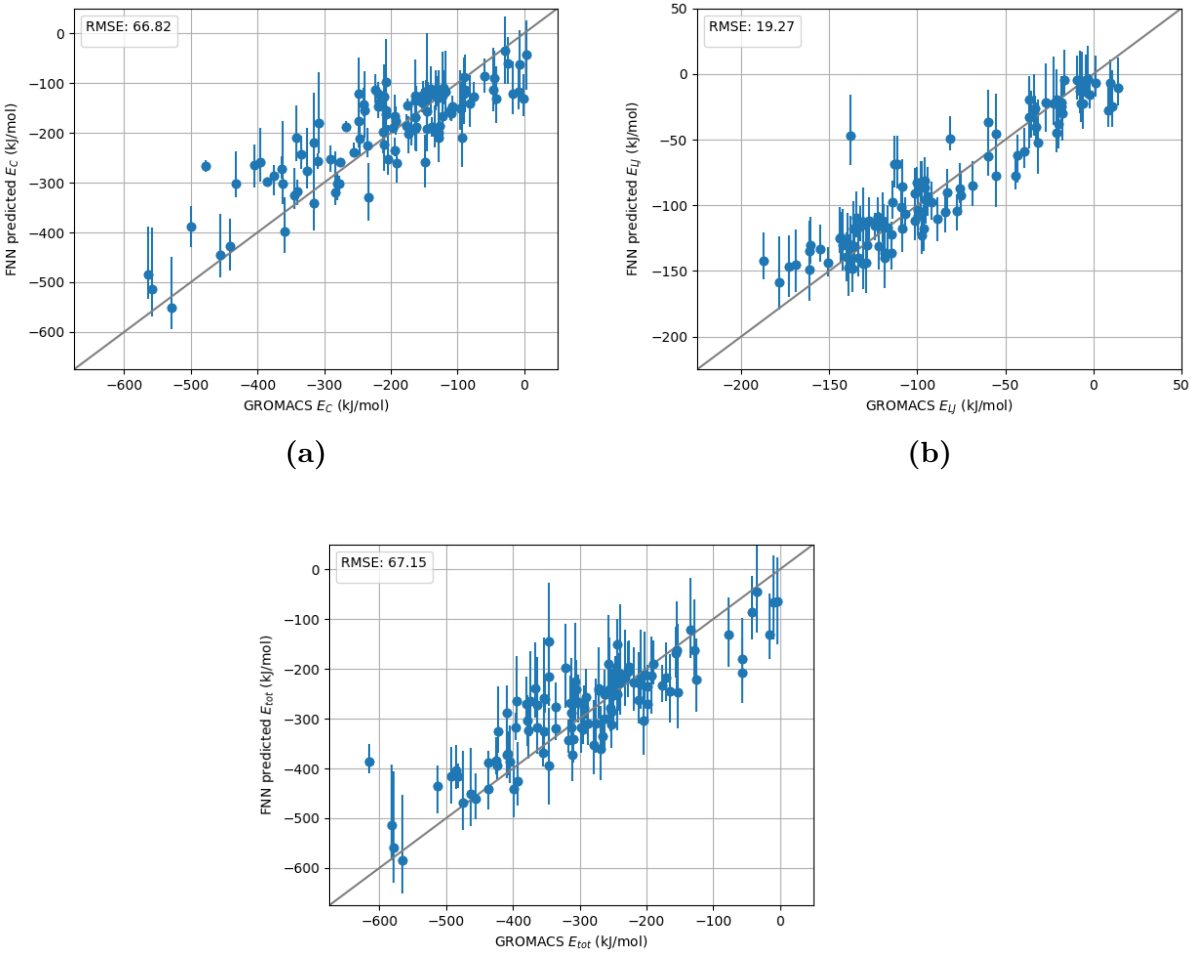
is harder to calculate accurately as described earlier when discussing force fields in section 2. The Lennard-Jones term is primarily dependent on atomic distances and it is more straightforward to calculate making it also easier to predict for the model. Proportionally the RMSE for the total energy, which is calculated by summing the Coulombic and Lennard-Jones terms, has a smaller error when compared to the terms separately. This could be because the model is not separately trained to predict the LJ and Coulombic terms, but the training for both is done at the same time.

The limitation when assessing the model performance using only the values provided in table 1 is that they do not measure the variance between the cross-validation models well. In figure 4 the predictions by the best performing model based on data from table 1 with two WL-updates on the data are plotted against the values from the GROMACS simulations. These predictions are made with the validation set left aside from cross validation. The data points are the mean values from the five cross validation models and the error limits are taken as the minimum and maximum values from the cross-validation predictions. The grey line in the plot indicates a perfect linear fit and points on the line correspond to exact precitions by the model. Based on these figures we can again see that the Lennard-Jones term is easier to predict for the model than the Coulombic term. Outliers are also clearly harder to predict which is generally expected for a machine learning model. In addition to the larger RMSE for the Coulombic term, the variation between the predictions based on the error limits in figure 4 varies proportionally more than for the Lennard-Jones term. Of course here the scale of the values has to be noted because the Coulombic term has values ranging from close to -600 kJ/mol to 0 kJ/mol, while the LJ term has a much smaller range of values.

Table 2. Averaged standard deviation from the predictions by the FNN model

Number of WL updates	Standard deviation σ_C (kJ/mol)	Standard deviation σ_{LJ} (kJ/mol)	Standard deviation σ_{tot} (kJ/mol)
1	34.04	11.63	29.18
2	**33.87**	**13.12**	**42.61**
3	34.46	7.73	35.83
4	30.70	10.36	31.38
5	44.75	14.46	47.63

To quantify this variance in the predictions from cross-validation the standard deviations for each of the energy terms and the total energy is presented in table 2. The standard deviations provided are calculated by calculating the standard deviation for each data point individually over predictions for all cross validation models and then averaging over these values. The average standard deviation between the predictions when training the model with different sets from cross-validation tells about the models ability to reliably generalise on unseen data. Based on the data from table 2 the model that was best based on data from table 1 has the second highest average standard deviation. However, the differences in the values between the models presented in 2 is not drastic considering the energy scales and the limited amount of data in training. The standard deviation on its own also doesn't quantify whether the predictions are right or wrong. The model could predict incorrect values with high confidence thus making for a small average standard deviation but subpar model performance. Thus it the model with two WL-updates, with bolded values also in table 2, on the data can still be considered to perform the best.



(a) **(b)**

(c)

Figure 4. Predicted values by the FNN model with two WL-updates on the data plotted against the GROMACS values used as validation. Datapoints are the mean from the five cross-validation models and error limits are taken from the minimum and maximum values from cross-validation. Points on the reference line (grey) correspond to exactly correct predictions. **a)** Results for the Coulombic interaction energy term. **b)** Results for the Lennard-Jones interaction energy term. **c)** Results for the total energy.

4.2 The custom GCN model

To study how the architecture of the custom GCN model affects the results multiple layer sizes and number of convolutions were tried for optimisation of the network. The results are depicted in Table 3 in the same fashion as for the FNN model. The layer sizes in the table depict the amount of neurons in the convolutional layers. One strategy in the layer architecture optimisation process was the keep the number of neurons the same, in this case 1280, while changing the amount of convolutions. In this process the features are transformed in to a higher-dimensional space from their original representation which can be helpful to capture more complex patterns in the data. The models trained based on this strategy all had 1280 neurons in total. Based on this process the model with three convolutions and convolutional layer sizes 512, 512 and 256 performed the best both in regards of RMSE and the Pearson coefficients. The values for this model are bolded in table 3.

Another optimisation strategy was to keep the output dimensions of the convolutional part the same as the input dimensions. As the part of the model consisting of convolutional layers operates straight on the protein graphs where each node has 39 features, the output dimensionality in this strategy was set to 39 while keeping the amount of neurons in the layers constant and changing the amount of convolutions. By keeping the dimensionality of the feature space the same as in the original representation makes to model simpler which could lead to more stable performance. For our model the performance is worse when matching the output and input dimensions than when the features space is transformed in to a higher dimension. This optimisation strategy was not optimal, as the poorest performing model from all the ones in table 3 is the one with every layer having an output dimension of 39. One reason for this could be that the model size doesn't allow for suitable learning capability. Another reason could be the fact that the model is built from two parts, the convolutions and the densely connected layer as presented in figure 3, and the amount of neurons in the first FNN layer at the end is 256. This means that by making the output from the convolutions smaller in dimension than the first FNN layer the layer size is truncated in the middle of the model. This may cause an information bottleneck because some information is lost and compressed between the two parts of the model. The exception to this is the model with layer sizes 1024 and

Table 3. RMSE and Pearson coefficients for the mean GCN predictions

Number of convolutions	Layer sizes	RMSE E_C (kJ/mol)	RMSE E_{LJ} (kJ/mol)	RMSE E_{tot} (kJ/mol)	Pearson r_C	Pearson r_{LJ}	Pearson r_{tot}
2	1024, 256	85.18	20.24	88.56	0.83	0.93	0.83
2	1024, 39	73.54	19.65	73.07	0.82	0.93	0.82
2	256, 256	77.71	21.23	80.36	0.82	0.93	0.82
3	**512, 512, 256**	**72.69**	**19.22**	**71.61**	**0.83**	**0.94**	**0.84**
3	512, 512, 39	82.05	19.65	81.39	0.81	0.93	0.82
3	256, 256, 256	77.38	20.68	78.68	0.82	0.93	0.83
3	39, 39, 39	80.47	22.10	80.97	0.78	0.92	0.79
4	512, 256, 256, 256	72.49	20.20	73.61	0.84	0.94	0.84
4	256, 256, 256, 256	76.78	23.88	82.71	0.83	0.93	0.83
4	512, 256, 256, 39	78.44	19.75	81.49	0.83	0.93	0.82

39 which may be due to the reduced complexity of the model when having fewer layers.

In table 4 the average standard deviations for the predictions from different cross-validation folds are presented in the same way as for the results for the FNN model with WL-updates presented earlier. The standard deviation is proportionally smaller for the total energy which makes sense as the same was the case for the RMSE and Pearson coefficient values. The reason for this again is probably the fact that the two non-bonded interaction terms are trained for and predicted at the same time. Considering the limited dataset size with the complexity of the custom GCN model in mind the standard deviations are within reasonable limits. Here there isn't a clear relationship between the average standard deviation for the predictions and the network size or the amount of convolutional layers. Thus it can still be said that the best performing model was the one with three convolutional layers with their respective sized being 512, 256 and 256 with bolded values in table 4. The results for this model are plotted in figure 5 with the predictions for the Coulombic term in figure 5a and for the Lennard-Jones term in 5c. The data points are again the mean values from the five predictions gotten from cross-validation and the error limits are taken as the minimum and maximum values from the cross-validation predictions. The grey line in the plot indicates a perfect linear fit. As expected the LJ-term has lower error limits and is easier to predict.

All in all from table 3 and table 4 it can be deduced that the best option is to

Table 4. Averaged standard deviation from the predictions by the GCN model

Number of convolutions	Layer sizes	Standard deviation σ_C (kJ/mol)	Standard deviation σ_{LJ} (kJ/mol)	Standard deviation σ_{tot} (kJ/mol)
2	1024, 256	43.57	12.96	54.63
2	1024, 39	30.46	11.17	34.95
2	256, 256	57.93	9.93	57.03
3	**512, 512, 256**	**33.42**	**7.87**	**28.87**
3	512, 512, 39	38.54	8.25	43.51
3	256, 256, 256	44.58	10.29	38.21
3	39, 39, 39	32.77	12.66	33.97
4	512, 256, 256, 256	48.90	12.24	49.86
4	256, 256, 256, 256	33.28	10.85	28.24
4	512, 256, 256, 39	36.96	5.78	37.64

keep the dimension of the residue representations higher than the size of the first dense layer in the FNN part of the model. Considering this I also employed and optimisation strategy where each layer is the size of 256 and the number of convolutions is altered. The motivation behind this was to keep the model as simple as possible while keeping the output dimension of the GCN part above the dimension of the first FNN layer. This did not yield better results than the first approach, thus implying increasing model complexity by adding neurons is favourable until a certain threshold. The models with three layers also generally have the best performance, if not the outlier of the model with the layer dimensions 1024 and 39, implicating that adding too many convolutions makes to model complexity too high for the data used. Overall the custom GCN model performed quite well with high linearity between the predictions and the validation values.

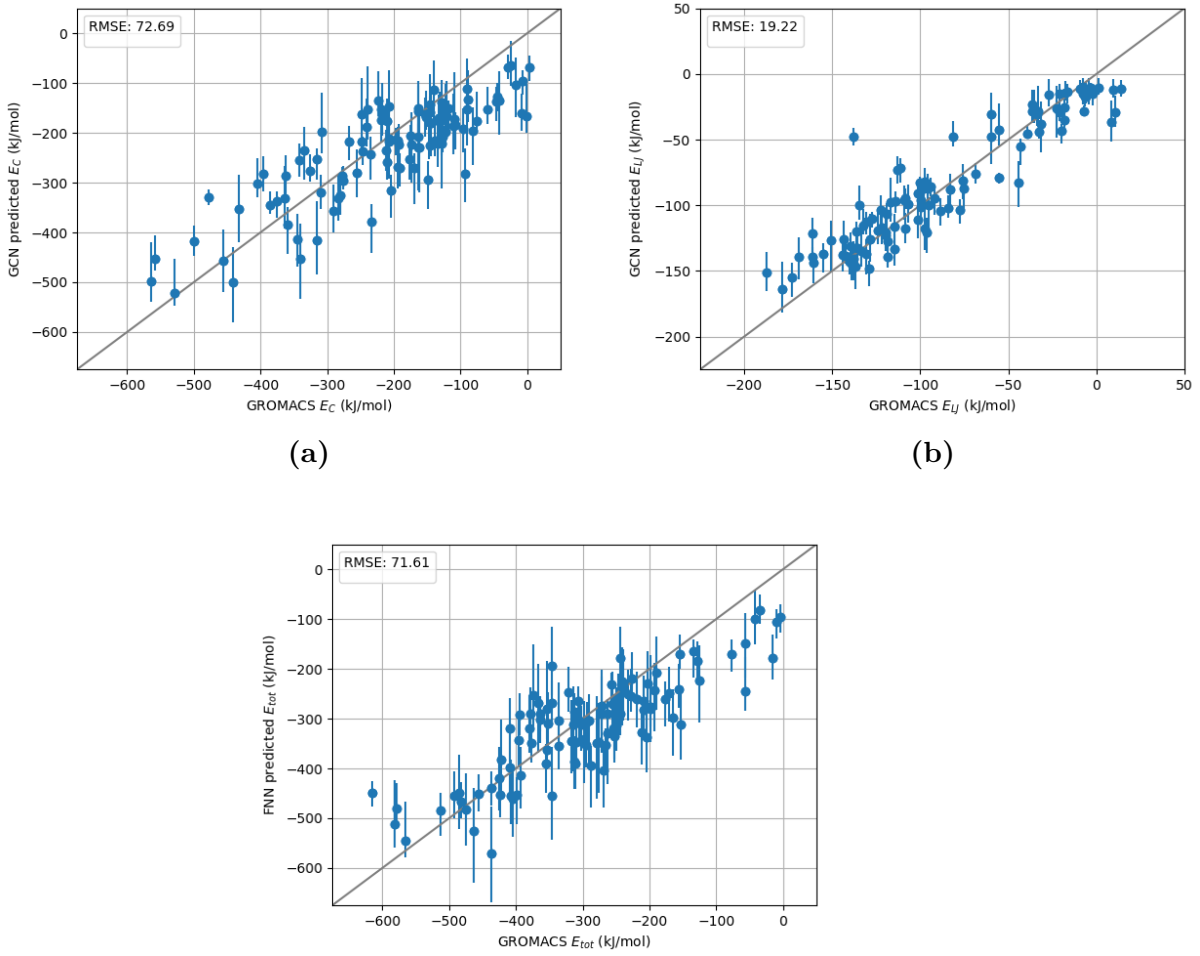


Figure 5. Predicted values by the GCN model with three convolutional layers with sizes 512, 512 and 256 plotted against the GROMACS values used as validation. Datapoints are the mean from the five cross-validation models and error limits are taken from the minimum and maximum values from cross-validation. Points on the reference line (grey) correspond to exactly correct predictions. **a)** Results for the Coulombic interaction energy term. **b)** Results for the Lennard-Jones interaction energy term. **c)** Results for the total energy.

4.3 Comparison of the models

Based on the results from tables 1 and 3 both model architectures show valid performance with a highly linear relationship between the predictions and target values. The most robust FNN model with two WL updates had slightly more accurate predictions than the GCN model with the best performance when comparing the Pearson coefficients and the RMSE values. This implies that the lower complexity and the focus on local information propagation in the graphs in the FNN combined with the WL-scheme is advantageous for this dataset. The limited size of the dataset may be a factor in this. The difference between the models is larger for the Coulombic term than for the LJ term. Being harder to predict, the Coulombic term can be expected to be the determining factor between the performance of the models.

As for the averaged standard deviations provided in table 2 and table 4 it can be seen that the values are quite similar for both architectures. From the standard deviations and the error limits in figure 4b and figure 5c it can be seen that the variance in predictions with the GCN model is lower for the Lennard-Jones term and the total energy than for the FNN model. For the Coulombic term such a general trend is not found. The most optimal GCN model from table 3 has lower variance in cross validation basen on the standard deviations than the most robust FNN model. The difference is noticeable especially for the total energy. This could imply that the complexity and expressiveness of graph convolutions leads to better generalisation on unseen data. Of course, it has to be noted that in terms of the RMSE and Pearson correlation the FNN model performed sligtly better. For both model architectures, the computational complexity didn't cause issues. This implies that the graph representation for the molecules is viable for these machine learning methods.

5 Conclusions and outlook

In this work a feed forward neural network combined with Weisfeiler-Lehman updates on graph form data and a custom graph convolutional network were successfully trained and used to predict non-bonded interactions energies between $Au_{25}(SR)_{18}$ nanoclusters and BSA proteins based on their features and data from molecular dynamics simulations. A graph representation for the proteins and the nanocluster-protein complexes was utilised for both models. The difference in the utilisation of graphs was that the GCN model operates straight on graph form data, while in the FNN model the graph representation is used in pre-processing to create a representation for the paired features. These paired features served as input to the FNN network. The dataset used was limited in size but still both models showed valid performance with linear relationship between the values predicted by the neural networks and the values provided from simulations. This shows that graph-based machine learning methods could be used to improve molecular dynamics simulations by creating a better initialization based on the interaction energies predicted by a neural network.

The focus of this thesis was on predicting the non-bonded interactions in the force fields used in MD simulations based on featurisation. The neural networks trained in this work worked well for predicting the non-bonded interaction energies between the nanocluster ligands and protein residues. These energies included the Lennard-Jones and Coulombic terms. The Coulombic term was harder to predict for both neural network architectures which is to be expected due to it's computational complexity and larger range of values. The RMSE for the Coulombic terms were proportionally larger in both cases and the linear correlation weaker based on Pearson correlation coefficients. The RMSE is also proportionally smaller than the errors for individual energy terms for both architectures. This is likely due to the fact that the models are trained for both terms simultaneously. One thing to keep in mind is also that the data used for training is from GROMACS simulations based on an empirical force fields. This means that while the results from work presented here can be used to improve the molecular dynamics simulations, generalisation of any quantitative

interactions parameters should be done with caution.

From the results alone it is hard to draw conclusions on which neural network architecture is better suited for the task. The best performance based on table 1 and table 3 was achieved with the FNN model using two WL-updates on the data but the difference was not drastic when compared to the best performing graph convolutional model. One thing to note from table 2 and 4 is that the most accurate custom GCN architecture has a smaller average standard deviation in cross-validation. Based on this the GCN model could generalise better on unseen data than the FNN model especially if optimised correctly. A significant advantage in the use of the GCN model is the fact that data in graph form can be used straight as input for the model thus requiring less pre-processing. The complexity of the GCN model is higher than that of the FNN model, which can be considered an asset or a disadvantage based on the application and the dataset used. In this work the dataset was limited in size thus favouring a more simple model. More work would have to be conducted with a larger dataset to draw better conclusions between the performance of the two models. In this work the computational complexity of the models didn't cause issues but if working with data where the graphs have a very large amount of edges, the GCN model may have issues with computational complexity. As mentioned earlier, the comparison between the two architectures is also interesting from a computer science perspective as there has previously been work on comparing graph convolutions to other graph propagation methods such as the Weisfeiler-Lehman scheme [25]. Graph convolutions have been proven to be at most as effective as the WL-scheme in labeling tasks and that result seems to apply also to the regression task in this work.

In this work early-stopping has been used to optimise the amount of epochs in the models and for the graph convolutional network the optimal architecture was searched for by adjusting layer sizes and the amount of convolutions. The learning rate was also adjusted for optimisation. More work would be needed to optimise the neural networks constructed in this work which would be better suited to be done when working on a larger dataset. More extensive hyperparameter tuning should be conducted if the neural networks architectures from this work would be turned in to reliable tools for molecular dynamics simulations. Different activation functions could also be tried even though ReLU seems to work well in the regression task

of predicting binding energies. For the feed forward neural network the amount of neurons and the amount of layers was not optimised because the emphasis of this work was on the graph convolutional part. The optimisation of the FNN model should be done before further work.

All in all, the results here have proven that the prediction of interaction energies between nanocluster and proteins is feasible using graph-based machine learning. Both neural network architectures showed valid perfomance in this task even though both of them have their distinct advantages and disadvantages. The implementation of these methods, with all of its practical considerations, to improve the initialization of molecular dynamics simulations is a possibility for future work. This could have implications, for example, in biomedical applications such as drug design.

References

[1] S. Kaptan and I. Vattulainen. "Machine learning in the analysis of biomolecular simulations". In: *Advances in Physics: X* 7.1 (2022), p. 2006080. DOI: 10.1080/23746149.2021.2006080. eprint: https://doi.org/10.1080/23746149.2021.2006080. URL: https://doi.org/10.1080/23746149.2021.2006080.

[2] I. Moscetti, S. Cannistraro, and A. R. Bizzarri. "Surface Plasmon Resonance Sensing of Biorecognition Interactions within the Tumor Suppressor p53 Network". In: *Sensors* 17.11 (2017). ISSN: 1424-8220. DOI: 10.3390/s17112680. URL: https://www.mdpi.com/1424-8220/17/11/2680.

[3] S. Y. Krisnakumar M. Ravikumar Wei Huang. "Coarse-Grained Simulations of Protein-Protein Association: An Energy Landscape Perspective". In: *Biophysical* 103.4 (2012), pp. 837–845. DOI: https://doi.org/10.1016/j.bpj.2012.07.013.

[4] M. E. Karpen, D. J. Tobias, and C. L. Brooks III. "Statistical clustering techniques for the analysis of long molecular dynamics trajectories: analysis of 2.2-ns trajectories of YPGDV". In: *Biochemistry* 32.2 (1993), pp. 412–420.

[5] M. Abraham et al. "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers". In: *SoftwareX* 1 (July 2015). DOI: 10.1016/j.softx.2015.06.001.

[6] M. Goldsmith et al. "Link Prediction with Continuous-Time Classical and Quantum Walks". In: *Entropy* 25.5 (2023). ISSN: 1099-4300. DOI: 10.3390/e25050730. URL: https://www.mdpi.com/1099-4300/25/5/730.

[7] M. Réau et al. "DeepRank-GNN: a graph neural network framework to learn patterns in protein–protein interfaces". In: *Bioinformatics* 39.1 (Nov. 2022), btac759. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btac759. eprint: https://academic.oup.com/bioinformatics/article-pdf/39/1/btac759/48448994/btac759.pdf. URL: https://doi.org/10.1093/bioinformatics/btac759.

[8] L. F. Krapp et al. “PeSTo: parameter-free geometric deep learning for accurate prediction of protein interacting interfaces”. In: *Nature Communications* 14 (2023). DOI: doi.org/10.1038/s41467-023-37701-8.

[9] H. M. Berman et al. “The Protein Data Bank”. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 235–242. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.235. eprint: https://academic.oup.com/nar/article-pdf/28/1/235/9895144/280235.pdf. URL: https://doi.org/10.1093/nar/28.1.235.

[10] M. Cha et al. “Unifying structural descriptors for biological and bioinspired nanoscale complexes”. In: *Nature Computational Science* 2 (2022), pp. 243–252. DOI: https://doi.org/10.1038/s43588-022-00229-w.

[11] T. Lengauer and M. Rarey. “Computational methods for biomolecular docking”. In: *Current Opinion in Structural Biology* 6.3 (1996), pp. 402–406. ISSN: 0959-440X. DOI: https://doi.org/10.1016/S0959-440X(96)80061-3. URL: https://www.sciencedirect.com/science/article/pii/S0959440X96800613.

[12] J. L. Morrison et al. “A lock-and-key model for protein–protein interactions”. In: *Bioinformatics* 22.16 (June 2006), pp. 2012–2019. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl338. eprint: https://academic.oup.com/bioinformatics/article-pdf/22/16/2012/48838808/bioinformatics_22_16_2012.pdf. URL: https://doi.org/10.1093/bioinformatics/btl338.

[13] A. Volkamer et al. “Combining Global and Local Measures for Structure-Based Druggability Predictions”. In: *Journal of Chemical Information and Modeling* 52.2 (2012). PMID: 22148551, pp. 360–372. DOI: 10.1021/ci200454v. eprint: https://doi.org/10.1021/ci200454v. URL: https://doi.org/10.1021/ci200454v.

[14] A. Pihlajamäki et al. “Graphs and Kernelized Learning Applied to Interactions of Hydrogen with Doped Gold Nanoparticle Electrocatalysts”. In: *The Journal of Physical Chemistry C* 127.29 (2023), pp. 14211–14221. DOI: 10.1021/acs.jpcc.3c02539. eprint: https://doi.org/10.1021/acs.jpcc.3c02539. URL: https://doi.org/10.1021/acs.jpcc.3c02539.

[15] A. Bujacz. “Structures of bovine, equine and leporine serum albumin”. In: *Acta Crystallographica Section D* 68.10 (Oct. 2012), pp. 1278–1289. DOI: 10.1107/S0907444912027047. URL: https://doi.org/10.1107/S0907444912027047.

[16] P. J. Theodore. *All About Albumin: Biochemistry, Genetics, and Medical Applications*. 1st ed. Academic Press, 1995. ISBN: 0125521103,9780125521109. URL: http://gen.lib.rus.ec/book/index.php?md5=cae4164b784d8e99f7dc66ee1803d693.

[17] H. Häkkinen and T. Tsukuda. *Protected metal clusters : from fundamentals to applications*. 1st ed. Frontiers of Nanoscience Volume 9. Elsevier, 2015. ISBN: 0081000863,978-0-08-100086-1,9780444635020,0444635025. URL: http://gen.lib.rus.ec/book/index.php?md5=3274143f75509fc212014ad2d8a30804.

[18] X. Kang, H. Chong, and M. Zhu. “Au25(SR)18: the captain of the great nanocluster ship”. In: *Nanoscale* 10 (23 2018), pp. 10758–10834. DOI: 10.1039/C8NR02973C. URL: http://dx.doi.org/10.1039/C8NR02973C.

[19] B. Zhang et al. “Ultrastable Hydrophilic Gold Nanoclusters Protected by Sulfonic Thiolate Ligands”. In: *The Journal of Physical Chemistry C* 125.1 (2021), pp. 489–497. DOI: 10.1021/acs.jpcc.0c08929. eprint: https://doi.org/10.1021/acs.jpcc.0c08929. URL: https://doi.org/10.1021/acs.jpcc.0c08929.

[20] B. Chakrabarty and N. Parekh. “NAPS: Network Analysis of Protein Structures”. In: *Nucleic Acids Research* 44.W1 (May 2016), W375–W382. ISSN: 0305-1048. DOI: 10.1093/nar/gkw383. eprint: https://academic.oup.com/nar/article-pdf/44/W1/W375/7634036/gkw383.pdf. URL: https://doi.org/10.1093/nar/gkw383.

[21] S. Patra and S. Vishveshwara. “Backbone cluster identification in proteins by a graph theoretical method”. In: *Biophysical chemistry* 84 (Mar. 2000), pp. 13–25. DOI: 10.1016/S0301-4622(99)00134-9.

[22] T. N. Kipf and M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG].

[23] N. Shervashidze et al. “Weisfeiler-Lehman Graph Kernels”. In: *Journal of Machine Learning Research* 12.77 (2011), pp. 2539–2561. URL: http://jmlr.org/papers/v12/shervashidze11a.html.

[24] L. Sai, L. Fu, and J. Zhao. "Predicting Binding Energies and Electronic Properties of Boron Nitride Fullerenes Using a Graph Convolutional Network". In: *Journal of Chemical Information and Modeling* 64.7 (2024). PMID: 38117935, pp. 2645–2653. DOI: 10.1021/acs.jcim.3c01708. eprint: https://doi.org/10.1021/acs.jcim.3c01708. URL: https://doi.org/10.1021/acs.jcim.3c01708.

[25] K. Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG].

[26] P. H. Hünenberger. "Thermostat Algorithms for Molecular Dynamics Simulations". In: *Advanced Computer Simulation: Approaches for Soft Matter Sciences I*. Ed. by C. Dr. Holm and K. Prof. Dr. Kremer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 105–149. ISBN: 978-3-540-31558-2. DOI: 10.1007/b99427. URL: https://doi.org/10.1007/b99427.

[27] M. P. Allen. *Computer simulation of liquids*. Ed. by D. J. Tildesley. Oxford science publications. Oxford : New York: Clarendon ; Oxford University Press, 1987. URL: https://jyu.finna.fi/Record/jykdok.320039.

[28] 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. eprint: https://pubs.aip.org/aip/jcp/article-pdf/81/8/3684/18950084/3684_1_online.pdf. URL: https://doi.org/10.1063/1.448118.

[29] B. Brooks, C. Brooks III, and A. M. J. et al. "CHARMM: The Biomolecular Simulation Program". In: *J. Comput. Chem.* 30 (2009). PMC2810661, 1545–1614.

[30] D. A. Case et al. "The Amber biomolecular simulation programs". In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1668–1688. DOI: https://doi.org/10.1002/jcc.20290. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20290. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20290.

[31] M. Christen et al. "The GROMOS software for biomolecular simulation: GROMOS05". In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: https://doi.org/10.1002/jcc.20303. eprint: https://

onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20303. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20303.

[32] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. "Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids". In: *Journal of the American Chemical Society* 118.45 (1996), pp. 11225–11236. DOI: 10.1021/ja9621760. eprint: https://doi.org/10.1021/ja9621760. URL: https://doi.org/10.1021/ja9621760.

[33] J. Lenhard, S. Stephan, and H. Hasse. "A Child of Prediction. On the History, Ontology, and Computation of the Lennard-Jonesium". In: *Studies in History and Philosophy of Science Part A* 103.C (2024), pp. 105–113. DOI: 10.1016/j.shpsa.2023.11.007.

[34] K. Vanommeslaeghe and A. MacKerell. "CHARMM additive and polarizable force fields for biophysics and computer-aided drug design". In: *Biochimica et Biophysica Acta (BBA) - General Subjects* 1850.5 (2015). Recent developments of molecular dynamics, pp. 861–871. ISSN: 0304-4165. DOI: https://doi.org/10.1016/j.bbagen.2014.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0304416514002736.

[35] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133. DOI: https://doi.org/10.1007/BF02478259.

[36] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark". In: *Neurocomputing* 503 (2022), pp. 92–108. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.06.111. URL: https://www.sciencedirect.com/science/article/pii/S0925231222008426.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[38] J. Han and M. Kamber. *Data Transformation and Data Discretization*. Elsevier, 2011. ISBN: 9780123814807.

[39] F. Chollet et al. *Keras*. https://keras.io. 2015.

[40] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.