Spring 2024

Predicting Biomolecular Properties and Interactions Using Numerical, Statistical and Machine Learning Methods

Elyssa Sliheet
Southern Methodist University, elyssasliheet@gmail.com

PREDICTING BIOMOLECULAR PROPERTIES AND INTERACTIONS USING NUMERICAL, STATISTICAL, AND MACHINE LEARNING METHODS

PREDICTING BIOMOLECULAR PROPERTIES AND INTERACTIONS USING NUMERICAL, STATISTICAL, AND MACHINE LEARNING METHODS

A Dissertation Presented to the Graduate Faculty of the

Dedman College

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computational and Applied Mathematics

by

Elyssa Nadine Sliheet

B.A., Mathematics, Southwestern University
B.A., Computer Science, Southwestern University
M.S., Computational and Applied Mathematics, Southern Methodist University

May 11, 2024

Acknowledgments

I would first like to acknowledge my advisor Dr. Weihua Geng, whose teaching, guidance, and mentorship shaped not only the quality of this dissertation but my overall experience as a graduate student. Thank you for your patience, wisdom, and encouragement.

I would like to thank my other committee members at Southern Methodist University, Dr. Andrea Barreiro and Dr. Wei Cai for their insightful feedback.

I am grateful to Dr. John Nemunaitis for providing me with the opportunity to work on real-world cancer applications. His commitment and dedication to putting patients first has inspired me.

I would like to thank Dr. Daniel Reynolds for his class on parallel scientific computing, which taught me the necessary skills for the high-performance computing work in this dissertation. All classes with Dr. Reynolds challenged me and made me a better programmer.

I appreciate all of the help from Rob Kalesky and John Lagrone from the SMU O'Donnell Data Science and Research Computing Institute for their eagerness to help me with all questions related to M2/M3.

I acknowledge the generous support from the Research Training Group (RTG) National Science Foundation (NSF) grant and Computational Science and Engineering fellowship funding sources that made this possible.

I would like to thank my friends at the EDGE (Enhancing Diversity in Graduate Education) program for offering me inspiration and a strong network of fellow women math warriors.

Heartfelt thanks to my dear friends, Molly Robinson and Sabrina Hetzel, for the countless early mornings and late nights studying for quals. Thank you for being amazing study buddies and even better friends. We did it.

I would like to thank my entire family but especially my parents, Bill and Jenny Sliheet. Thank you for paving the way. Thank you to my sisters, Sophia and Ava, for inspiring all that I do.

Lastly, I would like to thank my husband, Gregory O'Brien. Thank you for your endless love, support, and belief in me from the very beginning.

Sliheet, Elyssa Nadine B.A., Mathematics, Southwestern University
B.A., Computer Science, Southwestern University
M.S., Computational and Applied Mathematics, Southern Methodist University

Predicting Biomolecular Properties and Interactions
Using Numerical, Statistical, and Machine Learning Methods

Advisor: Dr. Weihua Geng
Doctor of Philosophy degree conferred May 11, 2024
Dissertation completed April 23, 2024

Machine learning models with their underlying numerical and statistical methods hold significant promise in various biological applications, particularly in predicting protein-ligand binding affinity, solvation energy, and patient clinical response to immunotherapy. These quantities help researchers better understand proteins and their interactions for disease prevention, immunization, and drug design. The work presented here focuses on designing mathematical models to extract valuable information from protein structural data both efficiently and accurately. We develop uniform feature extraction methods from both topological data analysis and electrostatics of charged proteins and protein-ligand complexes. In solving these mathematical models, we develop numerical methods to overcome challenges in accuracy and computational cost due to long-range and pairwise natures. In addition, a protein-protein interaction network is generated and studied to understand gene pair relationships relevant to patient clinical outcome. The contribution of this thesis toward the great computational biosciences community has the following three major components.

Parallel boundary integral PB solver

In our recent research in high-performance computing, we investigate the application of the Poisson-Boltzmann (PB) model in understanding the electrostatics of solvated biomolecules

relevant to the spread, treatment, and prevention of COVID-19. For each selected protein, the simulation produces the electrostatic solvation energy as a global measurement. We focus on parallelization strategies for the PB solver on central processing units (CPUs) versus graphical processing units (GPUs) and provide optimization guidance for selecting an appropriate computational method based on the size of the problem. Solving the PB model both rapidly and accurately is a critical step for the machine learning framework as it bridges biomolecular structures (the source of features) to its biological properties (the labels) obtainable with theoretic and computing approaches. This work was recently published in Computer Physics Communications [1].

Multiscale and uniform electrostatic and topological features extraction for machine learning

We describe our novel approach to design uniform scale-free electrostatic features as input to machine learning models for the prediction of Coulomb energy, solvation energy computed from the Poisson-Boltzmann model and solvation energy computed from the Generalized Born model. We further compare how these features as input impact model performance compared to topological features as input. Lastly, we investigate the approach of using both topological and electrostatic features in one cohesive model. These novel and efficient approaches to generate electrostatic and topological features can not only help to predict quantities associated with implicit solvation, but also have the potential to be used in general to represent the structural and associated force field information of the biomolecules to broader biophysical studies.

Models for immunotherapy in ovarian cancer

We also describe our work in collaboration with Gradalis, Inc. Despite the limited efficacy of immunotherapy in ovarian cancer, Vigil, a novel DNA-based immune therapeutic, has demonstrated clinical benefit to prolong relapse-free survival (RFS) and overall survival

(OS) in specific patient subgroups. Molecular analysis was conducted to identify biomarkers associated with sensitivity to Vigil treatment. Tissue samples from patients enrolled in a randomized double-blind trial of Vigil vs. placebo as maintenance in frontline management of advanced resectable ovarian cancer underwent DNA polymorphism analysis using a 981-gene panel. Pathogenic or likely pathogenic variants were analyzed and used to generate a protein-protein interaction network. We then calculated various metrics of the network to understand gene pair relationships. Kaplan-Meier analysis showed improved survival in Vigil-treated patients with specific genetic profiles. This exploratory study encourages further validation of Vigil efficacy in targeted patient populations and highlights the potential of network-based biomarker characterization in cancer immunotherapy research. This work was recently published in Cancer Gene Therapy [2].

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my little sisters, Sophia and Ava. You can do anything you set your mind to.

Chapter 1
Introduction

1.1. Research Motivation

Discovering the connections between protein structure and function is a very important task that lies at the intersection of biology, mathematics, and computer science. To make these connections, it is critical that we creatively extract and abstract information from proteins relevant to drug discovery and design. Important computational biophysics applications for machine learning models include the prediction of electrostatic solvation free energy, protein-ligand binding affinity, and clinical response of cancer patients who received immunotherapy. These quantities help researchers better understand proteins and their interactions for disease prevention, immunization, and drug design. With this, the complex structure of biomolecules can limit the promise of powerful models. Our project aims to design mathematical models that can be solved with efficient and accurate numerical algorithms to ultimately discover the useful information that is hidden in complex protein structural data.

Specifically, we incorporate physics-informed features in topology *and* electrostatics. In this work, we aim to design a machine learning model that utilizes biologically relevant information from the fields of topology and electrostatics with the hypothesis that combining these ideas into one cohesive model will improve model performance and demonstrate the need to leverage the combination of different mathematical techniques to predict important biological quantities. The topological features abstract the underlying atomic relationships and patterns to generate image-like representations that can utilize the advancements of convolutional neural networks. This approach has shown success on the task of predicting protein-ligand binding affinity. We take this idea further and apply it to the task of predicting solvation energy while incorporating physical considerations with our designed electrostatic features to improve performance even further.

While doing so, we consider the balance between computational efficiency and accuracy. As researchers, we should take advantage of the availability of large datasets from sources such as the Protein Data Bank (PDB) [3] while also designing methods that are efficient when dealing with large datasets of large biomolecules.

Additionally, we study the Poisson-Boltzmann (PB) model that incorporates quantities such as media permittivity, protein charge distribution, electrolyte distribution in solvent, temperature, etc. With the assistance of numerical algorithms and computational tools, the PB model has broad applications in biomolecular simulations, including protein structure [4], protein-protein interaction [5, 6], chromatin packing [7], protein pKa values, protein-membrane interactions, [8, 9], binding energy [10, 11], solvation free energy [12, 13], and ion channel profiling [14].

Solving the PB model accurately and efficiently is challenging due to a variety of factors, such as the large problem dimension, the complex geometry of the protein, the jump of dielectric constants across media interface, and the singular charge representation. We focus on the parallelization development of boundary integral PB solvers.

More specifically, we investigate two approaches for the parallelization of boundary integral PB solvers. One is the parallelization of the treecode-accelerated boundary integral (TABI) solver using MPI which builds an identical tree on each task/CPU. Its parallelization occurs at four stages of the TABI solver: source term computation, treecode for matrix-vector product, preconditioning, and energy computation. We apply the schemes developed for n-body parallelization [15] to a more complicated boundary integral PB problem, and develop MPI-based parallelization for the preconditioning scheme designed for boundary integral solvers [16]. Our second parallelization approach focuses on GPU-based parallelization of the direct-sum boundary integral (DSBI) solver, which concurrently computes the source term, matrix-vector product, and energy computation. We provide guidance for users when the DSBI solver on GPU or the TABI solver with MPI on CPUs should be used depending on the size of the problem.

Lastly, we describe our work which is independent of the above projects to computationally identify biomarkers from clinical trial and patient genetic data acquired from Gradalis, Inc. The trial data involved patients enrolled in a randomized double-blind trial of Vigil vs. placebo as maintenance in frontline management of advanced resectable ovarian cancer. Ovarian cancer is the third most common gynecologic cancer, and it carries the worst prognosis and highest mortality rate of gynecologic cancers [17–19]. Mortality from ovarian cancer is three times that of breast cancer [19, 20]. Genetically, the majority of ovarian cancer cases are *BRCA* wild type (*BRCA*wt) but more than one-fifth of cases are attributable to mutations in tumor suppressor genes, with 65–85% of the mutations being in germline *BRCA* genes (g*BRCA*) [21, 22].

Standard treatment of resectable newly diagnosed stage III/IV ovarian cancer involves surgical resection and adjuvant or neoadjuvant chemotherapy [23, 24]. Unfortunately, nearly 75% of this patient population who undergo standard treatment will experience recurrence following frontline therapy [23, 25]. The establishment of more effective therapies for ovarian cancer is essential.

Vigil is an autologous tumor DNA immunotherapy which was designed to enhance the immune system's potency against cancer in three ways: first, Vigil introduces the individual tumor neoantigen repertoire to the immune system. Second, Vigil enhances differentiation and activation of immune cells via GM-CSF, a cytokine important to immune activation at both the peripheral and marrow levels. Finally, Vigil inhibits cancer expressive TGF-beta, thereby decreasing immunosuppressive activity of TGF-beta.

We hypothesize that intact DNA repair mechanisms of *BRCA*wt, HRP ovarian cancer may be important for Vigil efficacy, possibly related to higher degree of clonal versus subclonal neoantigens available for anticancer immune stimulation [26, 27].

We describe further molecular analysis in coordination with clinical benefit parameters of genomic variant data in all patients involved in the VITAL trial. We seek to identify significant genomic variants, meaningful variant combinations, and relevant genes at the intersection or "hub" of ovarian cancer pathways which provide proof of principle to a novel clinically applicable method of biomarker assessment.

1.2. Outline

The remainder of this thesis is organized as follows, Chapter 2 describes both the biological and mathematical background including the Poisson-Boltzmann (PB) and Generalized Born (GB) models. Chapter 3 discusses the development of the parallelization techniques used. Chapter 3 also describes the feature extraction methods used to generate the topological and electrostatic features and the machine learning framework. Chapter 4 reports the results of the parallelization methods on various COVID-19 proteins and machine learning performance metrics using our novel approach to produce electrostatic features alone and in combination with the topological features. Chapter 5 describes our previous work in collaboration with Gradalis Inc. to identify biomarkers for patients who received novel Vigil immunotherapy. We summarize our work and include concluding remarks in Chapter 6. We offer software dissemination details in Chapter 7.

Chapter 2

Background

In this chapter, we give biological and mathematical background related to the proposed research. In the first section, we provide a background of the relevant biological information and define the specific quantities we aim to predict. Additionally, we describe the mathematical foundation of the computational models we will investigate and develop in the next chapter.

2.1. Biological Background

This project studies the biophysics of proteins. To this end, we provide some biological background involving protein structures, solvation energy, and binding energy. Particularly, we will introduce proteins relating to COVID-19 which are the focus of the electrostatic analysis and parallelization development.

2.1.1. Protein Structure

To study proteins, it is important to first describe basic structural information. Deoxyribonucleic acid (DNA) is transcribed to ribonucleic acid (RNA), where a gene provides the code to synthesize messenger RNA (mRNA). Next, during translation, mRNA determines the order of amino acids, which then build proteins. The most simple level of protein structure is primary structure, which consists of sequences of amino acid chains. The next level of protein structure is secondary, which has two types; α-helices and β-sheets which are characterized by hydrogen bonds between the main-chain peptide groups/backbone. The next level, tertiary structure, is the 3D structure of the protein where the α-helices and β-sheets are folded into one compact structure. Tertiary structure refers to one single polypeptide chain. The final, most complex, level is quaternary, which is the 3D structure consisting of more than one polypeptide chain. To analyze proteins computationally, we export data from the Protein Data Bank [3], which consists of over 200,000 biological macromolecular structures of proteins. This vast dataset was cultivated experimentally, and users have the option to select relevant data based on organism type, taxonomy, experimental method, enzyme classification type, etc. This abundant dataset enables breakthroughs in research and it is critical to develop mathematical methods to accurately and efficiently produce biological insights from this data. More specifically, the data we use falls into the following file types:

- `.pdb` files: The data is from the Protein Data Bank [3]. Each protein has a corresponding `.pdb` file that provides the sequences of amino acids in the peptide chains, coordinates of the atoms in three dimensions, atom types and radius of each atom.

- `.pqr` files: In order to run our simulations, the partial charge of each atom is needed. We use a software called pdb2pqr [28] to convert the `.pdb` files to `.pqr` files which contain the partial charges. This is done by incorporating a user-specified force field. Force field refers to the functional form and parameter sets used to calculate the potential energy of a system of atoms or coarse-grained particles in molecular mechanics and molecular dynamics simulations. The parameters of the energy functions can be derived from experimental work and quantum mechanical calculations. We choose to use AMBER for the force field.

- `.mol2` files: For the task of predicting protein-ligand binding affinity we need a computational representation of the ligand files, which are represented by `.mol2` files. To convert these files to `.pqr` files we use a software called Open Babel [29].

2.1.2. Solvation Energy

The first task we aim to predict is solvation energy. Proteins are solvated, that is, they are surrounded by a solvent, typically taken to be water. The interaction between the molecules of the protein and the solvent molecules is called solvation, which is critical for protein structural dynamics. Solvation energy is governed by the interactions between the solute and solvent molecules, including electrostatic interactions, hydrogen bonds, van der Waals forces, and other intermolecular forces. Our focus in this thesis is the electrostatic component of the solvation energy, i.e. electrostatic solvation energy. For simplicity, hereafter, when the term ”solvation energy” is used, we refer to the electrostatic solvation energy. We can compute the solvation energy of a protein with both the Poisson-Boltzmann and Generalized Born models. The Poisson-Boltzmann model provides higher accuracy at the cost of increased computational complexity, while the Generalized Born model is more computationally efficient but less accurate.

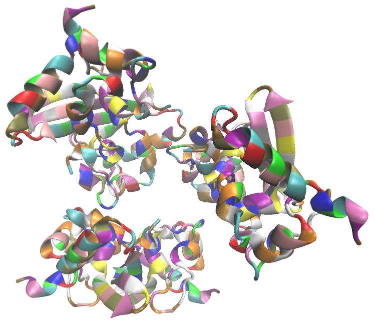
2.1.3. Binding Affinity

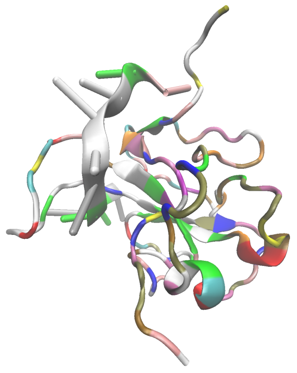
Binding affinity is defined as a measure of the strength of the binding interaction between a biomolecule to its ligand/binding partner. Binding affinity is typically measured by the equilibrium dissociation constant (KD) [30]. The smaller the KD value, the greater the binding affinity of the ligand for its target. The larger the KD value, the weaker the biomolecule and ligand are attracted to and bind to one another. Binding affinity is influenced by many factors including hydrogen bonding, electrostatic interactions, hydrophobic and Van der Waals forces between the two molecules as well as the presence of other molecules. The advancement of methods for the prediction of binding affinity between various proteins and ligands is essential to understanding intermolecular interactions. These interactions are important factors driving biological processes and this task is critical for drug discovery and design.

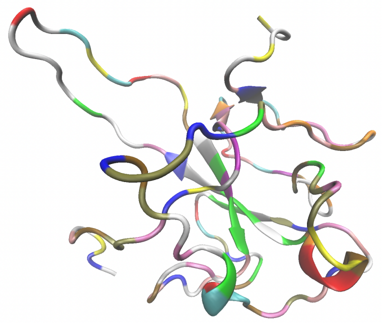
2.1.4. COVID-19 Proteins

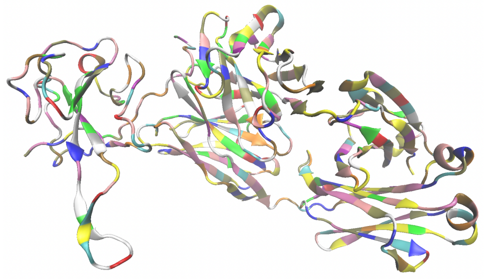
In Chapter 4, we present results on parallelization schemes to solve the Poisson Boltzmann model on various proteins relevant to COVID-19. Coronaviruses are a persistent threat to global health. Viruses such as SARS in 2003, MERS in 2013, and the new SARS-CoV-2 in 2019 emerge from animal populations and can then infect humans. Coronaviruses contain a large genome that directs the synthesis of several dozen viral proteins. Structures of these proteins are used to better understand the diseases and to develop new drugs and vaccines to fight coronaviruses. We focus on proteins involved in the spreading and prevention of the COVID-19 virus. The virus genome in the form of an mRNA encodes proteins including replication/transcription complexes that make more RNA, structural proteins that construct new virions, and proteases (e.g. 61u7) that cut polyproteins into all of these functional pieces. The virus docks to target cells by binding the spike protein (e.g. proteins 6crz, 6vxx, 6vsp, 6vsb) on the viral surface to its receptor, angiotensin-converting enzyme 2 (ACE2, e.g. protein 6m17) on the target cell membrane. In addition, to test the infection of COVID-19 virus, we often identify its nucleocapsid proteins (e.g. proteins 7act, 6yi3) by using antibodies (e.g. proteins 7cr5, 7n3c, 7sts) that particularly bind to these nucleocapsids [3]. In this work, we select a few COVID-19-related proteins and use our parallel PB solvers to calculate their electrostatic properties such as global solvation energy or local surface potential. These protein electrostatics can assist researchers in understanding a protein's overall structure and function, their binding affinity to certain ligands, as well as their folding and enzyme catalysis characteristics. Consequently, efficient algorithms employing cutting-edge techniques such as parallelization, are vital for researchers working on vaccine and drug development.

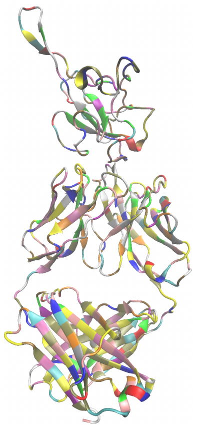
In Figure 2.1, we provide the cartoon structure of six COVID-19-related proteins. Protein 6wji [31] in (a) is a dimerization domain, which is used to bring two nucleocapsids together. The connections of nucleocapsid dimers into bigger groups makes the viral structure that encases the RNA in the limited area within virus particles. The SARS-CoV-2 nucleocapsid contains separate proteins which all perform different functions. A portion of the structure folds into an RNA-binding domain (protein 7act) [32] as shown in (b), featuring a groove that securely holds a brief segment of the viral genomic RNA. In contrast, the protein alone without the RNA-bound structure (protein 6yi3) is shown in (c) [32]. In COVID-19 prevention, home test kits for detecting SARS-CoV-2 infection rely on antibody proteins that specifically recognize nucleocapsids within a complex set of biomolecules in nasal samples. The antibodies they recognize differ based on the test-kit brand, which recognizes different portions of the nucleocapsid, and we list a few here with protein 7cr5 [33] in (d), protein 7n3c in (e), and protein 7sts in (f). For these listed proteins, the PB model is solved using our parallel boundary integral PB solvers, and numerical results are presented in Chapter 4.


(a)


(b)


(c)


(e)


(f)

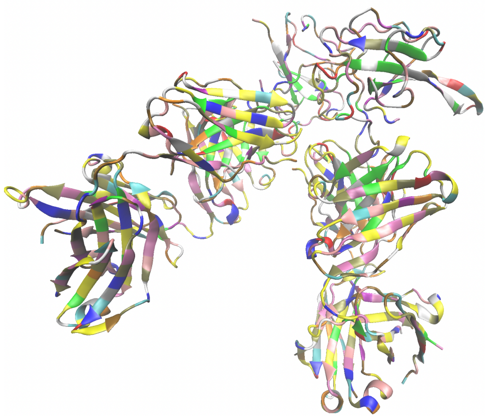

(g)

Figure 2.1: COVID-19 related proteins used in our numerical simulations. (a) **6wji**: The C-terminal Dimerization Domain of Nucleocapsid Phosphoprotein from SARS-CoV-2; (b) **7act**: SARS-CoV-2 Nucleocapsid Phosphoprotein N-Terminal Domain in Complex with 10mer ssRNA; (c) **6yi3**: The N-terminal RNA-Binding Domain of the SARS-CoV-2 Nucleocapsid Phosphoprotein; (d) **7cr5**: Human Monoclonal Antibody with SARS-CoV-2 Nucleocapsid Protein NTD; (e)**7n3c**: Human Fab S24-202 in the Complex with the N-Terminal Domain of Nucleocapsid Protein from SARS-CoV-2; (f) **7sts**: Human Fab S24-1379 in the Complex with the N-terminal Domain of Nucleocapsid Protein from SARS-CoV-2.

2.2. Mathematical Background

In order to approach the tasks of predicting solvation energy and binding affinity, we need a basic mathematical understanding of the underlying biophysical properties of the proteins in question. We start with a basic theory of electrostatics, followed by the two popular implicit solvent models, the Poisson-Boltzmann model and the Generalized Born model.

2.2.1. Introduction to Electrostatics

We introduce electrostatics by starting with its role in molecular dynamics simulations, which obeys Newton's Second Law:

$$\vec{F} = ma \tag{2.1}$$

where $\vec{F}$, m and a denote force, mass, and acceleration, respectively. For each configuration, the potential energy, E, is given by,

$$E = E_{\text{bond}} + E_{\text{ang}} + E_{\text{tors}} + E_{\text{vdw}} + E_{\text{elec}} \tag{2.2}$$

where E_{bond} is the energy due to bond stretching, E_{ang} is the energy due to angle bending, E_{tors} is the energy due to bond rotation (torsion), E_{vdw} is the Van der Waals interaction which is modeled by the Lennard-Jones potential and E_{elec} is the electrostatic energy, given by Coulomb's law. Note that electrostatic interactions are long-range and pairwise, thus computationally expensive to compute. This is the reason why computing electrostatic interactions attracts the attention of mathematicians.

Potentials and fields due to electrical charges are given by,

$$\vec{F} = \frac{ee'}{r^2}\frac{\vec{r}}{r} \tag{2.3}$$

which is Coulomb's Law in vector form, and r is the distance between the source charge e and the evaluation point e'. The electric field is given by the limit of the electric force $\vec{F}$ per unit charge e',

$$\vec{E} = \lim_{e' \to 0} \frac{\vec{F}}{e'} = \frac{e}{r^2} \frac{\vec{r}}{r} \tag{2.4}$$

Now for a closed surface S, and da an element of surface area, we have

$$\oint_S \vec{E} \cdot \vec{n} \, \mathrm{d}a = 4\pi \sum_i e_i \tag{2.5}$$

where $\vec{n}$ is the outward unit normal to the surface, by the principle of superposition. For a continuous charge density, $\rho(x)$, we have

$$\oint_S \vec{E} \cdot \vec{n} \, \mathrm{d}a = 4\pi \int_\Omega \rho(x) \, \mathrm{d}^3 x \tag{2.6}$$

Now apply the divergence theorem to the left-hand side of Equation 2.6 to arrive at,

$$\int_\Omega \nabla \cdot \vec{E} \, \mathrm{d}^3 x = 4\pi \int_\Omega \rho(x) \, \mathrm{d}^3 x \tag{2.7}$$

where Ω is the volume surrounded by the closed surface S, which leads to Gauss's Law:

$$\nabla \cdot \vec{E} = \rho \tag{2.8}$$

. Recall that a vector field can be specified up to a constant if its curl and divergence are given everywhere in space. The curl of $\vec{E}$ is equal to zero under the electrostatic assumption of Faraday's Law, that is $\partial_t \vec{H} = 0$, where $\vec{H}$ is the magnetic field. Since $\nabla \times \vec{E} = 0$, $\vec{E}$ can be written as the negative gradient of a scalar potential, ϕ. Plugging this into Gauss's Law (2.8), we have,

$$-\nabla \cdot \nabla \phi = \rho \tag{2.9}$$

which is the Poisson Equation [34].

The Poisson model is the governing equation for the electrostatic potential subject to the given charge distribution. When considering the complexities of charge distribution within the solute and solvent domains, the Poisson-Boltzmann model offers a more accurate framework. In practice, the solvent can be simulative with water molecules with atomic details explicitly. To reduce the computational cost, we adopt the implicit solvent model.

2.2.2. Implicit Solvent Model

To study solvation energy, we operate under the implicit solvent framework. As opposed to the explicit solvent model which describes the system at the atomic level and is computationally demanding, the implicit solvent framework assumes that the solvent is approximated as a continuum and ions are described by some statistical distribution. These assumptions greatly reduce the computational complexity of the problem of calculating the electrostatic potential. Here, we introduce two popular implicit solvent models, the Poisson Boltzmann and Generalized Born models. Both govern the electrostatics of the target protein, with varying levels of accuracy and efficiency.

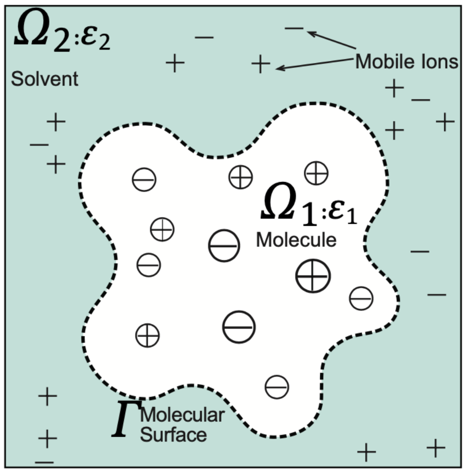

$\Omega_2 : \varepsilon_2$
Solvent
Mobile Ions
$\Omega_1 : \varepsilon_1$
Molecule
Γ Molecular Surface

Figure 2.2: Implicit solvent framework computational domain: Ω_1 (dielectric constant ϵ_1 with partial charges as a weighted summation of delta functions) and Ω_2 (dielectric constant ϵ_2 with mobile ions modeled by Boltzmann distribution) are separated by the molecular surface Γ.

2.2.2.1. The Poisson-Boltzmann Model

The Poisson-Boltzman Equation arises in cases where the charge density ρ depends on the potential ϕ. The Boltzmann distribution law states that the ratio of the concentration of one type of ion near the molecule to its concentration far from the molecule is given by

$$e^{-W_i(\mathbf{x})/kT} \tag{2.10}$$

where $W_i(\mathbf{x})$ is the work required to move the ion of type i from $|\mathbf{x}|= \infty$ to the point $\mathbf{x}$, T is the temperature and k is the Boltzmann constant. As seen in Figure 2.2, the types of mobile ions are positive and negative, so we have

$$W_1(\mathbf{x}) = +e_c\phi(\mathbf{x}) \tag{2.11}$$

and

$$W_2(\mathbf{x}) = -e_c\phi(\mathbf{x}) \tag{2.12}$$

where e_c is the charge of an electron. Let M denote the bulk concentration of ions per cubic centimeter for each of two ions with charge $+e_c$ and $-e_c$. Now by the Boltzmann distribution law, we have,

$$M_+ = Me^{-e_c\phi(\mathbf{x})/kT} \tag{2.13}$$

and

$$M_- = Me^{+e_c\phi(\mathbf{x})/kT} \tag{2.14}$$

So the charge density is given by,

$$\rho(x) = M_+ e_c - M_- e_c \tag{2.15}$$

$$= M e^{-e_c \phi(\mathbf{x})/kT} e_c - M e^{+e_c \phi(\mathbf{x})/kT} e_c \tag{2.16}$$

$$= -2 M e_c \text{sinh} \left(\frac{e_c \phi(\mathbf{x})}{kT} \right) \tag{2.17}$$

Plugging ρ from Equation 2.17 into the Poisson equation (Equation 2.9) results in the non-linear Poisson-Boltzmann equation,

$$\Delta \phi = 2 M e_c \text{sinh} \left(\frac{e_c \phi(\mathbf{x})}{kT} \right) \tag{2.18}$$

The linearized Poisson-Boltzmann equation is derived by taking the first term in the series expansion of

$$\text{sinh} x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots \tag{2.19}$$

So then the linear Poisson Boltzmann equation is given by,

$$\nabla^2 \phi_1(\mathbf{x}) = -4\pi \sum_{i=1}^{N} q_i \delta(\mathbf{x} - \mathbf{x}_k), \quad \mathbf{x} \in \Omega_1 \tag{2.20}$$

$$\nabla^2 \phi_2(\mathbf{x}) = \kappa^2 \phi_2(\mathbf{x}), \quad \mathbf{x} \in \Omega_2 \tag{2.21}$$

where Ω_1 is the region of the domain inside of the molecular surface and Ω_2 is the exterior of the molecular surface. The Poisson-Boltzmann model is shown in Figure 2.2, where the molecular surface Γ divides the entire computational domain Ω into the protein domain Ω_1 with dielectric constant ϵ_1 and atomic charges q_k located at $\mathbf{x}_k, k = 1 : N_c$, and the solvent domain Ω_2 with dielectric constant ϵ_2 and dissolved salt ions. And κ denotes the *Debye-Huckel* parameter defined to be

$$\kappa = \left(\frac{2 M e_c^2}{kT} \right)^{1/2} \tag{2.22}$$

Additionally, the interface conditions on the molecular surface, Γ, are given by,

$$\phi_1(\mathbf{x}) = \phi_2(\mathbf{x}), \tag{2.23}$$

$$\epsilon_1 \frac{\partial \phi_1(\mathbf{x})}{\partial \mathbf{n}} = \epsilon_2 \frac{\partial \phi_2(\mathbf{x})}{\partial \mathbf{n}}, \tag{2.24}$$

$$\lim_{|\mathbf{x}| \to \infty} \phi_2(\mathbf{x}) = 0 \tag{2.25}$$

[35, 36]. The PB model governs the electrostatic potential ϕ in the entire space. Theoretically, after ϕ is obtained, its gradient will produce the electrostatic field while its integral will generate potential energy. However, there are many challenging issues on properly obtaining the field and energy (e.g. definition of field on molecular surface Γ [37]). Our attention for this project is on the energy as described below. The electrostatic potential energy is given as

$$E = \frac{1}{2}\int_{\Omega} \rho(\mathbf{x})\phi(\mathbf{x})d\mathbf{x} = \frac{1}{2}\sum_{k=1}^{N_c} q_k \phi(\mathbf{x}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k(\phi_{\text{reac}}(\mathbf{x}_k) + \phi_{\text{coul}}(\mathbf{x}_k)) = E_{\text{solv}} + E_{\text{coul}} \tag{2.26}$$

where $\rho(\mathbf{x}) = \sum\limits_{\mathbf{k=1}}^{\mathbf{N_c}} \mathbf{q_k}(\mathbf{x} - \mathbf{x_k})$ is the charge density as a sum of partial charges weighted delta function and the $E_{\text{solv}} = \frac{1}{2}\sum\limits_{k=1}^{N_c} q_k \phi_{\text{reac}}(\mathbf{x}_k)$ term is the solvation energy, the energy it takes for the protein to solvate from the vacuum to the solvent. The ϕ_{reac} is the reaction potential as the remaining component when Coulomb potential ϕ_{coul} is taken away from the total electrostatic potential ϕ.

2.2.2.2. The Generalized Born Model

Now the Generalized Born model approximates the electrostatic part of the solvation free energy. To derive the Generalized Born model, we first adopt some classic results for electrostatics from Jackson's textbook [34]. Using the analog from discrete point charge distribution to continuous charge density distribution, the potential energy of a charged system with density distribution $\rho(\mathbf{r})$ takes the form

$$W = \frac{1}{2}\int \rho(\mathbf{r})\phi(\mathbf{r})d\mathbf{r} = \frac{1}{2}\int \vec{E}(\mathbf{r}) \cdot \vec{D}(\mathbf{r})d\mathbf{r}$$

where the electric field $\vec{E} = -\nabla\phi$, and electric displacement $\vec{D} = \epsilon\vec{E}$.

Now consider assembling a charge to the center at the origin of a sphere with radius r_i. The sphere separates the domain with ϵ_{in} for $r < a_i$ and ϵ_{out} for $r < a_i$. The assembly takes the energy

$$G_i = \frac{1}{8\pi}\int \frac{\vec{D}\cdot\vec{D}}{\epsilon}d\vec{r} \approx \frac{1}{8\pi}\int_{r<a_i} \frac{q_i^2}{r^4\epsilon_{\text{in}}}dx + \frac{1}{8\pi}\int_{r>a_i} \frac{q_i^2}{r^4\epsilon_{\text{out}}}dx \tag{2.27}$$

where the Coulomb field approximation $\vec{D} = \frac{q\mathbf{r}}{r^3}$ is used.
The solvation energy is the energy difference when ϵ for $r > a_i$ changes from ϵ_{in} (unsolvated) to ϵ_{out} (solvated) given as

$$\Delta G_{\text{solv},i} = \frac{1}{8\pi}(\frac{1}{\epsilon_{\text{out}}} - \frac{1}{\epsilon_{\text{in}}})\int_{r>a_i} \frac{q_i^2}{r^4}dx = (\frac{1}{\epsilon_{\text{out}}} - \frac{1}{\epsilon_{\text{in}}})\frac{q_i^2}{2a_i}. \tag{2.28}$$

This result is consist with the Poisson-Boltzmann model of a solvated spherical cavity with centered charges as summarized in [36], which is a special case from Kirkwood's derivation of a series of spherical harmonics for a spherical cavity containing arbitrary multiple charges [38]. If we treat the molecule as a collection of spherical atoms, using Equation (2.28) the total solvation energy is

$$\Delta G_{\text{solv}} = \frac{1}{2}\left(\frac{1}{\epsilon_{\text{out}}} - \frac{1}{\epsilon_{\text{in}}}\right)\left(\sum_{i=1}^{N}\frac{q_i^2}{a_i} + \sum_{j\neq i}^{N}\frac{q_i q_j}{r_{ij}}\right) \approx \frac{1}{2}(\frac{1}{\epsilon_{\text{out}}} - \frac{1}{\epsilon_{\text{in}}})\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{f_{ij}^{\text{GB}}} \tag{2.29}$$

where the first equation has two terms: a sum of individual Born terms and pairwise Coulombic terms. Note the *actual* molecule is better represented by a dielectric interface (e.g. solvent excluded surface (SES)) which separates inside domain Ω_{in} and outside domain Ω_{out}. The charge q_i is located at the center $\mathbf{r}_i$ of a sphere with radius a_i. We assume Ω_{in} contains all these spheres. Thus the model using dielectric interface requires a step further as been approximated in the second equation in Equation (2.29).

Here f_{ij}^{GB} is the effective Born radii ($i = j$) or effective interaction distance ($i \neq j$). Assuming the Born Radii R_i's are obtained, a popular estimation of f_{ij}^{GB} is:

$$f_{ij}^{\text{GB}}(r_{ij}) = (r_{ij}^2 + R_i R_j e^{\frac{-r_{ij}^2}{4R_i R_j}})^{1/2}$$

where r_{ij} is the distance between the atomic centers of atoms i and j. Note R_i depends not only on a_i, but also on radii and relative positions of all other atoms. The estimation of effective Born radii is an active research area. From Poisson-Boltzmann theory, the *perfect* Born radii is given as

$$R_i = \frac{1}{2}(\frac{1}{\epsilon_{\text{out}}} - \frac{1}{\epsilon_{\text{in}}})\frac{q_i}{\Delta G_{\text{PB},i}}$$

with $\Delta G_{\text{PB},i}$ as the solvation energy from the PB model with all except ith charge muted. There are many approaches to approximate R_i and a typical one is the volume integration via quadrature as

$$R_i^{-1} = \frac{1}{4\pi}\int_{\Omega_{\text{out}}}\frac{q_i^2}{r^4}d\mathbf{r} = \frac{q_i^2}{a_i} - \frac{1}{4\pi}\int_{\Omega_{\text{in}}/B(\mathbf{r}_i,a_i)}\frac{q_i^2}{r^4}d\mathbf{r}.$$

Computing R_i using FFT leads to the computational cost of sovaltion energy to $O(n^3 \log n + N + N^2)$ [39] against the cost of solving the PB model of $O(n^6 + Nn^2 + N)$ with finite difference method (iterative solver, matrix structure not considered), where n is the number of grid point in x,y,and z direction, assuming cube-alike computational domain.

Now that we have established the background on the Poisson-Boltzmann model and Generalized Born, we will now further discuss how to computationally solve the Poisson-Boltzmann model, including parallelization techniques. Then, how we curate topological and electrostatic features to be combined to ultimately predict biomolecular properties.

Chapter 3
Numerical Algorithms

In this chapter, we detail the steps that are used to solve the Poisson-Boltzmann equation described in Chapter 2 as well as our methodology to parallelize the solver using both CPUs and GPU. Additionally, we describe how we generate multi-scale and uniform electrostatic features. Then, we outline the machine learning methods which utilize the aforementioned topological and electrostatic features, including details of model architecture and design.

3.1. Numerical Methods to Solve the Poisson Boltzmann Equation

Solving the Poisson Boltzmann model numerically is challenging due to various reasons including

1. the protein is represented by singular point charges
2. the molecular surface is geometrically complex
3. the dielectric constant is discontinuous across the surface and
4. the domain is unbounded

To overcome these numerical difficulties, we utilize boundary element methods (BEM) for the PB model [40–45] which have several inherent advantages,

1. only the molecular surface is discretized rather than the entire solute/solvent volume
2. the atomic charges are treated analytically

3. the interface conditions are accurately enforced
4. the far-field boundary condition is imposed analytically

In the original BEMs, these advantages were offset by the high cost of evaluating the interactions among the elements, but fast summation schemes have been developed to reduce the cost [43,45–49]. For example, we can employ the treecode algorithm to reduce the computational cost associated with each iteration of solving the resulting linear system. In developing these boundary integral PB solvers, many interesting numerical challenges arise, e.g. preconditioning of matrix whose conditioner number increases when triangulation quality is reduced, the parallelization of the treecode algorithm using MPI [15] and the parallelization of the boundary integral PB solver using GPU [50, 51]. The developed solvers help us to produce electrostatic potentials, which can be further used to compute protein properties such as binding energy to ligands [11].

In summary, the major steps to solve the PB model are as follows,

1. Triangulate the molecular surface using an established surface generator software.
2. Use the boundary element method to get a set of coupled integral equations.
3. Discretize integral equations with centroid collocation to arrive at a linear system.
4. Solve the linear system with GMRES using the treecode algorithm to speed up matrix-vector product calculation.

Then we can compute the electrostatic solvation energy from the resulting potential.

We present the boundary integral form of the PB implicit solvent model, the discretization of the boundary integral equations, the treecode algorithm for accelerating the matrix-vector product, and the preconditioning scheme to alleviate the rising condition number when the triangulation quality deteriorates due to complex geometry [16, 52, 53].

3.1.1. Boundary Integral Form of Poisson-Boltzmann Model

Following the tradition of the boundary integral method, we use $\mathbf{x}$ and $\mathbf{y}$ to represent the spatial position. We also denote $\Omega_1 = \Omega^-$, $\Omega_2 = \Omega^+$, $\epsilon_1 = \epsilon^-$, and $\epsilon_2 = \epsilon^+$. This section summarizes the well-conditioned boundary integral form of the PB implicit solvent model we employ [41, 53]. Applying Green's second identity and properties of fundamental solutions to Eq. (2.20) yields the electrostatic potential in each domain,

$$\phi(\mathbf{x}) = \int_\Gamma \left[G_0(\mathbf{x},\mathbf{y}) \frac{\partial \phi(\mathbf{y})}{\partial \nu} - \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}} \phi(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}} + \sum_{k=1}^{N_c} q_k G_0(\mathbf{x},\mathbf{y}_k), \quad \mathbf{x} \in \Omega_1, \tag{3.1a}$$

$$\phi(\mathbf{x}) = \int_\Gamma \left[-G_\kappa(\mathbf{x},\mathbf{y}) \frac{\partial \phi(\mathbf{y})}{\partial \nu} + \frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}} \phi(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}}, \quad \mathbf{x} \in \Omega_2, \tag{3.1b}$$

where $G_0(\mathbf{x},\mathbf{y})$ and $G_\kappa(\mathbf{x},\mathbf{y})$ are the Coulomb and screened Coulomb potentials, respectively

$$G_0(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi|\mathbf{x}-\mathbf{y}|} \quad \text{and} \quad G_\kappa(\mathbf{x},\mathbf{y}) = \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}. \tag{3.2}$$

Then applying the interface conditions in Eq. (2.23) with the differentiation of electrostatic potential in each domain yields a set of boundary integral equations relating the surface potential ϕ_1 and its normal derivative $\partial\phi_1/\partial\nu$ on Γ,

$$\frac{1}{2}(1+\varepsilon)\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x},\mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}} + S_1(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{3.3a}$$

$$\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right) \frac{\partial \phi_1(\mathbf{x})}{\partial \nu} = \int_\Gamma \left[K_3(\mathbf{x},\mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_4(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}} + S_2(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{3.3b}$$

where $\varepsilon = \varepsilon_2/\varepsilon_1$. As given in Eqs. (3.4a-3.4b) and (3.8), the kernels $K_{1,2,3,4}$ and source terms $S_{1,2}$ are linear combinations of G_0, G_k, and their first and second order normal derivatives [41, 53],

$$K_1(\mathbf{x},\mathbf{y}) = G_0(\mathbf{x},\mathbf{y}) - G_\kappa(\mathbf{x},\mathbf{y}), \quad K_2(\mathbf{x},\mathbf{y}) = \varepsilon \frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}} - \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}}, \tag{3.4a}$$

$$K_3(\mathbf{x},\mathbf{y}) = \frac{\partial G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}} - \frac{1}{\varepsilon}\frac{\partial G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}}, \quad K_4(\mathbf{x},\mathbf{y}) = \frac{\partial^2 G_\kappa(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}\partial \nu_{\mathbf{y}}} - \frac{\partial^2 G_0(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}\partial \nu_{\mathbf{y}}}, \tag{3.4b}$$

where the normal derivative with respect to $\mathbf{x}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{x}}} = -\nu(\mathbf{x}) \cdot \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) = -\sum_{m=1}^{3} \nu_m(\mathbf{x}) \partial_{x_m} G(\mathbf{x},\mathbf{y}), \tag{3.5}$$

the normal derivative with respect to $\mathbf{y}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}} = \nu(\mathbf{y}) \cdot \nabla_{\mathbf{y}} G(\mathbf{x},\mathbf{y}) = \sum_{n=1}^{3} \nu_n(\mathbf{y}) \partial_{y_n} G(\mathbf{x},\mathbf{y}), \tag{3.6}$$

the second normal derivative with respect to $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \nu_{\mathbf{y}}\partial \nu_{\mathbf{x}}} = -\sum_{m=1}^{3}\sum_{n=1}^{3} \nu_m(\mathbf{x})\nu_n(\mathbf{y}) \partial_{x_m}\partial_{y_n} G(\mathbf{x},\mathbf{y}), \tag{3.7}$$

and the source terms $S_{1,2}$ are

$$S_1(\mathbf{x}) = \frac{1}{\varepsilon_1} \sum_{k=1}^{N_c} q_k G_0(\mathbf{x},\mathbf{y}_k) \quad \text{and} \quad S_2(\mathbf{x}) = \frac{1}{\varepsilon_1} \sum_{k=1}^{N_c} q_k \frac{\partial G_0(\mathbf{x},\mathbf{y}_k)}{\partial \nu_{\mathbf{x}}}. \tag{3.8}$$

Once the potential and its normal derivative are solved from Eqs. (3.3a)-(3.3b), the potential at any point inside the molecule can be computed via Eq. (3.1a), or a numerically more accurate formulation may be used from [41]:

$$\phi_1(\mathbf{x}) = \int_\Gamma \left[K_1(\mathbf{x},\mathbf{y}) \frac{\partial \phi_1(\mathbf{y})}{\partial \nu} + K_2(\mathbf{x},\mathbf{y})\phi_1(\mathbf{y}) \right] \mathrm{d}S_{\mathbf{y}} + S_1(\mathbf{x}), \quad \mathbf{x} \in \Omega_1. \tag{3.9}$$

With the potential and its normal derivative on Γ, the electrostatic free energy can be obtained by

$$E^{\text{free}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k\phi_1(\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k \left(\int_\Gamma \left[K_1(\mathbf{y}_k,\mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_2(\mathbf{y}_k,\mathbf{y})\phi_1(\mathbf{y})\right]\mathrm{d}S_{\mathbf{y}} + S_1(\mathbf{y}_k)\right). \tag{3.10}$$

The electrostatic solvation energy can also be obtained by

$$E^{\text{sol}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k\phi^{\text{reac}}(\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{N_c} q_k \int_\Gamma \left[K_1(\mathbf{y}_k,\mathbf{y})\frac{\partial\phi_1(\mathbf{y})}{\partial\nu} + K_2(\mathbf{y}_k,\mathbf{y})\phi_1(\mathbf{y})\right]\mathrm{d}S_{\mathbf{y}}, \tag{3.11}$$

where $\phi^{\text{reac}}(\mathbf{x}) = \phi(\mathbf{x}) - S_1(\mathbf{x})$ is the reaction field potential [41,53].

3.1.2. Discretization of Boundary Integral Equations

The integrals in Eqs. (3.3a)-(3.3b) can be discretized by centroid collocation, which is popular due to its simplicity [53]. Alternatively, it can be discretized using more complicated approaches such as node collocation [54,55], curved triangles [56], or Galerkin's method [57], with each resulting in a trade-off between accuracy and efficiency. Here we employ the centroid collocation approach.

Letting $\mathbf{x}_i, i = 1,\ldots,N$ denote the triangle centroids of the N triangular elements, the discretized Eqs. (3.3a)-(3.3b) have the following form for $i = 1,\ldots,N$,

$$\frac{1}{2}(1+\varepsilon)\,\phi_1(\mathbf{x}_i) = \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_1(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu} + K_2(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]\Delta s_j + S_1(\mathbf{x}_i), \tag{3.12a}$$

$$\frac{1}{2}\left(1+\frac{1}{\varepsilon}\right)\frac{\partial\phi_1(\mathbf{x}_i)}{\partial\nu} = \sum_{\substack{j=1\\ j\neq i}}^{N}\left[K_3(\mathbf{x}_i,\mathbf{x}_j)\frac{\partial\phi_1(\mathbf{x}_j)}{\partial\nu} + K_4(\mathbf{x}_i,\mathbf{x}_j)\phi_1(\mathbf{x}_j)\right]\Delta s_j + S_2(\mathbf{x}_i), \tag{3.12b}$$

where Δs_j is the area of the jth boundary element, and the term $j = i$ is omitted from the summation to avoid the kernel singularity. Eqs. (3.12a)-(3.12b) represent a linear system $Ax = b$, where $\mathbf{x}$ contains the surface potentials $\phi_1(\mathbf{x}_i)$ and normal derivatives $\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \nu}$, weighted by the element area Δs_i, and $\mathbf{b}$ contains the source terms $S_1(\mathbf{x}_i)$ and $S_2(\mathbf{x}_i)$. We solve this system using the generalized minimal residual (GMRES) iterative method, which requires a matrix-vector product in each step [58]. Since the matrix is dense, computing the product by direct summation requires $O(n^2)$ operations, which is prohibitively expensive when n is large. These difficulties can be overcome by fast algorithms for n-body computations, such as treecode [51, 53] and Fast Multipole Methods [45, 57]. In the next section, we describe how the treecode algorithm is used to accelerate the matrix-vector product calculation.

3.1.3. Treecode

We summarize the treecode algorithm and refer to previous work for more details [46, 52, 59, 60]. The matrix-vector product Ax for Eqs. (3.12a)-(3.12b) has the form of n-body potentials,

$$V_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} q_j G(\mathbf{x}_i, \mathbf{x}_j), \quad i = 1, \ldots, N, \tag{3.13}$$

where G is a kernel, $\mathbf{x}_i, \mathbf{x}_j$ are centroids (also called particle locations in this context), and q_j is a charge associated with $\mathbf{x}_j$. To this end, the q_j in Eq. (3.13) is equivalent to the $\Delta s_j \phi_1(\mathbf{x}_j)$ or $\Delta s_j \frac{\partial \phi_1(\mathbf{x}_j)}{\partial \nu}$ in Eqs. (3.12a)-(3.12b) and G is one of the kernels K_{1-4}. To evaluate the potentials V_i rapidly, the particles $\mathbf{x}_i$ are divided into a hierarchy of clusters having a tree structure in a 2-D illustration as in Fig. 3.1(a). The root cluster is a cube containing all the particles and subsequent levels are obtained by dividing a parent cluster into children [46]. The process continues until a cluster has fewer than N_0 particles (N_0 is a user-specified parameter representing the maximum number of particles per leaf, e.g. $N_0 = 3$ in Fig. 3.1(a)). Then V_i is evaluated as a sum of particle-particle interactions and particle-cluster interactions (depicted in Fig. 3.1(b)),

$$V_i \approx \sum_{c\in N_i}\sum_{\mathbf{x}_j\in c} q_j G(\mathbf{x}_i,\mathbf{x}_j) + \sum_{c\in F_i}\sum_{\|\mathbf{k}\|=0}^{p} a^{\mathbf{k}}(\mathbf{x}_i,\mathbf{x}_c)\, m_c^{\mathbf{k}}, \tag{3.14}$$

where c denotes a cluster, and N_i, F_i denote the near-field and far-field clusters of particle $\mathbf{x}_i$. The first term on the right is a direct sum for particles $\mathbf{x}_j$ near $\mathbf{x}_i$, and the second term is a pth order Cartesian Taylor approximation about the cluster center $\mathbf{x}_c$ for clusters that are well-separated from $\mathbf{x}_i$ [52]. The Taylor coefficients are given by

$$a^{\mathbf{k}}(\mathbf{x}_i,\mathbf{x}_c) = \frac{1}{\mathbf{k}!}\partial_{\mathbf{y}}^{\mathbf{k}} G(\mathbf{x}_i,\mathbf{x}_c), \tag{3.15}$$

and the cluster moments are given by

$$m_c^{\mathbf{k}} = \sum_{\mathbf{x}_j\in c} q_j(\mathbf{x}_j-\mathbf{x}_c)^{\mathbf{k}}. \tag{3.16}$$

Cartesian multi-index notation is used with $\mathbf{k} = (k_1,k_2,k_3), k_i \in \mathbb{N}$ and $\|\mathbf{k}\| = k_1+k_2+k_3$. A particle $\mathbf{x}_i$ and a cluster c are defined to be well-separated if the multipole acceptance criterion (MAC) is satisfied, $r_c/R \le \theta$, where r_c is the cluster radius, $R = |\mathbf{x}_i - \mathbf{x}_c|$ is the particle-cluster distance and θ is a user-specified parameter [46].

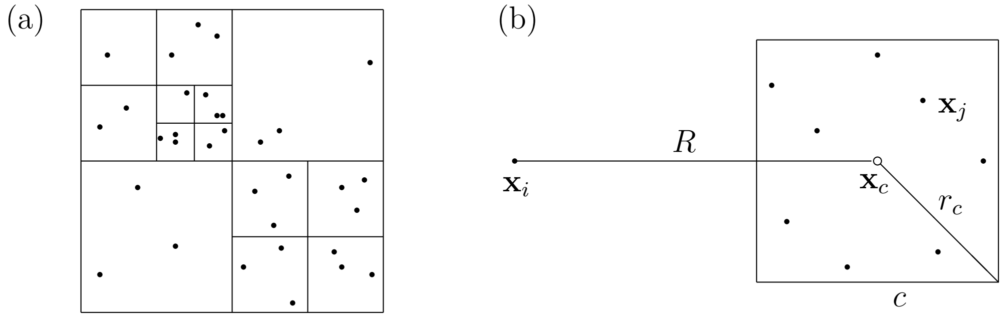


Figure 3.1: Details of treecode. (a) tree structure of particle clusters. (b) particle-cluster interaction between particle $\mathbf{x}_i$ and cluster $c = \{\mathbf{x}_j\}$. $\mathbf{x}_c$: cluster center; R: particle-cluster distance; and r_c: cluster radius.

The accuracy and efficiency of the treecode is controlled by the combination of parameters including the order p, MAC parameter θ, and maximum particles per leaf N_0. Using the treecode, the operation count for the matrix-vector product is $O(N \log N)$, where N is the number of particles $\mathbf{x}_i$, and the factor $\log N$ is the number of levels in the tree.

3.1.4. Preconditioning

In order to precondition Krylov subspace methods in solving $Ax = b$, we design a scheme using left-preconditioning. Given a preconditioning matrix M, we consider the modified linear system $M^{-1}Ax = M^{-1}b$. The solve then proceeds in two steps: (a) set $c = M^{-1}b$, and (b) solve $(M^{-1}A)x = c$ using GMRES. We therefore seek a preconditioner, M, such that two conditions are satisfied:
(1) M is similar to A such that $M^{-1}A$ has improved condition compared to A and requires fewer GMRES iterations;
(2) $M^{-1}z = y$ can be efficiently computed, which is equivalent to solving y from $My = z$.
Conditions (1) and (2) cannot be improved concurrently, thus a trade-off must be made.

We designf our preconditioner based on the observation that in the electrostatic interactions, which is also the interactions between boundary elements in solving integral equations, the short range interactions are smaller in number of interactions, but more significant in strength than the long range interactions, which are large in number of interactions and computationally more expensive. Due to their large number of interactions, the long interactions are calculated by multipole expansions. This offers the idea that for a preconditioner of A, we may construct M to contain only short range interactions and to ignore long range interactions. To this end, we select short range interactions between elements on the same leaf only. This choice of M has great advantages in efficiency and accuracy for solving $My = z$. As seen with details in [16], by using a permutation operation, the M matrix is a block diagonal matrix with n blocks such that $M = \text{diag}\{M_1, M_2, \cdots, M_n\}$ thus $My = z$ can be solved using direct method e.g., LU factorization by solving each individual $M_i y_i = z_i$ for $i = 1, \cdots, n$. Here each M_i is a square nonsingular matrix, which represents the interaction between particles/elements on the ith leaf of the tree. It is worthy to note that the efficiency is not affected even when M_i has a large condition number since a direct solver is used for solving $My = z$. Meanwhile, the computational cost for solving $My = z$ is $O(Nn_0^2)$ with n_0 the treecode parameter, maximum number of particles per leaf, as detailed in [16].

3.2. Parallelization

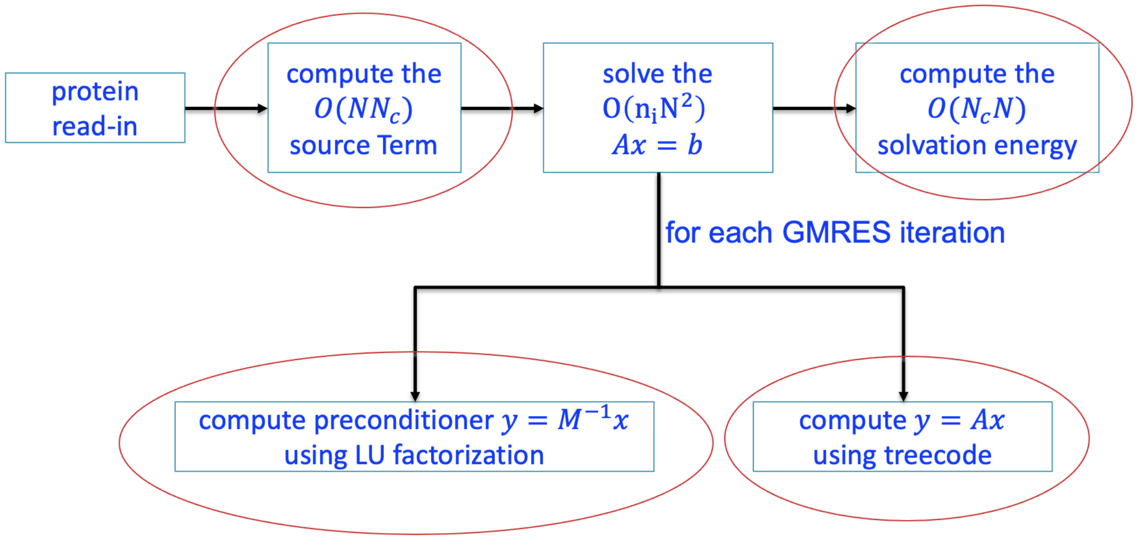

protein read-in
compute the $O(NN_c)$ source Term
solve the $O(n_i N^2)$ $Ax = b$
compute the $O(N_c N)$ solvation energy
for each GMRES iteration
compute preconditioner $y = M^{-1}x$ using LU factorization
compute $y = Ax$ using treecode

Figure 3.2: pipeline for parallelized TABI solver

3.2.1. MPI-based Parallelization of the TABI Solver

As illustrated by the red circled items in Fig. 3.2, our parallelization of the TABI solver focuses on the four stages of the pipeline: the $O(NN_c)$ source term, $O(n_i N \log N)$ matrix-vector product using treecode, the $O(n_i N N_0^2)$ preconditioner, and the $O(N_c N)$ solvation energy. Here, N is the number of surface triangles, N_0 is the maximum number of particles per leaf, N_c is number of partial charges, and n_i is the number of GMRES iterations. Among these stages, the most time consuming and challenging component is the matrix-vector product using treecode. To this end, we investigate two possible strategies for computing n-body problems as in [61], which are also briefly described below. In this work, we migrate these strategies to solving the boundary integral PB model. The numerical results in Chapter 4 show high parallel efficiency from the optimized approach.

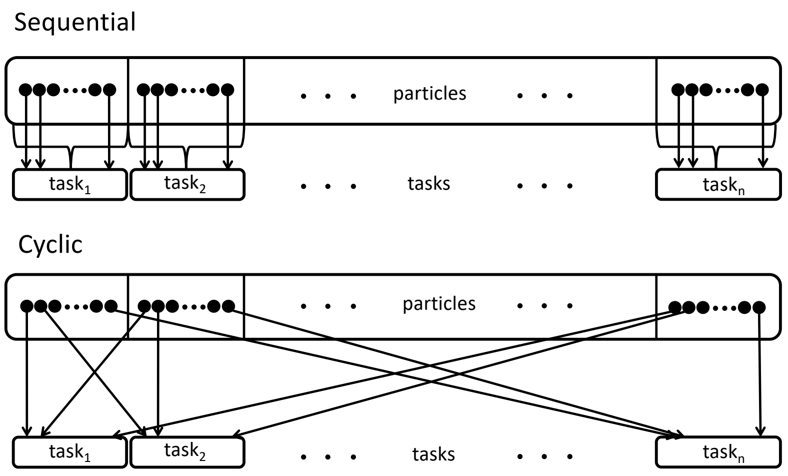


Figure 3.3: methods for assigning target particles to tasks: sequential order (top) vs cyclic order (bottom)

The initial and intuitive method to assign target particles to tasks is to use *sequential ordering*, in which the 1st task handles the first N/n_p particles in a consecutive segment, the 2nd task handles the next N/n_p particles, etc. The illustration of this job assignment is shown in the top of Fig. 3.3. However, when examining the resulting CPU time on each task, we noticed starkly different times on each task, indicating a severe load imbalance. This may be understood by the fact that for particles at different locations, the types of interactions with the other particles through the tree can vary. For example, a particle with only a few close neighbors uses more particle-cluster interactions than particle-particle interactions, thus requiring less CPU time than a particle with many close neighbors. We also notice that for particles that are nearby one another, their interactions with other particles, either by particle-particle interaction or particle-cluster interaction, are quite similar, so some consecutive segments ended up computing many more particle-particle interactions than others that were instead dominated by particle-cluster interactions. Based on these observations, we designed a *cyclic ordering* scheme to improve load balancing, as illustrated on the bottom of Fig. 3.3. In this scheme, particles nearby one another are uniformly distributed to different tasks. For example, for a group of particles close to each other, the first particle is handled by the first task, the second particle is handled by the second task, etc. The cycle repeats starting from the $(n_p + 1)$-th particle. The numerical results that follow demonstrate the significantly improved load balance from this simple scheme.

The pseudocode for our MPI-based parallel TABI solver using replicated data algorithm is given in Table 3.1. The identical trees are built on each task as in line 6. The four-stage MPI-based parallelization of the source term, matrix-vector product with treecode, preconditioning, and solvation energy occur in lines 4, 10, 12, and 17, respectively, followed by MPI communications.

3.2.2. The GPU-accelerated DSBI Solver

The pseudocode for the DSBI-PB solver using GPUs is given in Table 3.2. In this psuedocode, we divide all the operations into those on host performed by the CPUs and those on device performed by the GPUs. The three compute-intensive stages are computation of the source term, matrix-vector product, and solvation energy; each are computed on GPUs as shown in lines 5, 10, and 19, followed by a copy of the data from device to host. The host CPU takes care of all complicated and non-concurrent work. We note that lines 13 and 14 are still under investigation due to the considerations of parallel efficiency, and we disable these two lines in our current numerical implementation. The challenge is that the variance in sizes of the block matrices M_i for $i = 1, \ldots, n$ that compose the preconditioner M lead to significant load imbalance on the GPU. However, disabling the preconditioner within the GPU based parallelization could significantly increase computing time when A is ill-conditioned.

Table 3.1: Pseudocode for MPI-based parallel TABI solver using replicated data algorithm.

```
 1  on main processor
 2      read protein data
 3      call MSMS to generate triangulation
 4      copy protein data and triangulation to all other processors
 5  on each processor
 6      build local copy of tree
 7      compute assigned segment of source terms by direct sum
 8          copy result to all other processors
 9      set initial guess for GMRES iteration
10      compute assigned segment of matrix-vector product $Ax$ by treecode
11          copy result to all other processors
12      compute assigned segment of solving $Mx = y$ for $x$ by LU factorization .
13          copy result to all other processors
14      test for GMRES convergence
15          if no, go to step 10 for next iteration
16          if yes, go to step 15
17      compute assigned segment of electrostatic solvation energy by direct sum
18          copy result to main processor
19  on main processor
20      add segments of electrostatic solvation energy and output result
```

Table 3.2: Pseudocode for DSBI-PB solver using GPU

```
  1  On host (CPU)
  2      read biomolecule data (charge and structure)
  3      call MSMS to generate triangulation
  4      copy biomolecule data and triangulation to device
  5  On device (GPU)
  6      each thread concurrently computes and stores source terms
  7      copy source terms on device to host
  8  On host
  9      set initial guess $\mathbf{x}_0$ for GMRES iteration and copy it to device
 10  On device
 11      each thread concurrently computes assigned segment of $\mathbf{y} = \mathbf{A}\mathbf{x}$
 12      copy the computed matrix-vector $\mathbf{y}$ to host memory
13*      each thread concurrently solves its assigned portion of $\mathbf{M}\mathbf{x} = \mathbf{y}$
14*      copy the solution $\mathbf{x}$ to host memory
 15  On host
 16      test for GMRES convergence
 17          if no, generate new $\mathbf{x}$ and copy it to device, go to step 10
 18          if yes, generate and copy the final solution to device, go to step 19
 19  On device
 20      compute assigned segment of electrostatic solvation energy
 21      copy computed electrostatic solvation energy contributions to host
 22  On host
 23      add segments of electrostatic solvation energy and output result
     * currently disabled
```

3.3. Uniform Scale-free Electrostatic Features

In majority of the molecular simulations, including Monte Carlo simulation, Brownian Dynamics, Molecular Dynamics, etc, the electrostatic interactions are characterized by the interactions between the partial charges assigned at the atomic centers by the force field, which are determined by experiment or quantum chemistry. To reduce the computational cost, implicit solvent model such as Poisson-Boltzmann model and Generalized Born model are used in which the water is treated as a continuum while partial charges are still assigned at the atomic center of the solute (the protein). Our consideration of electrostatic interactions is under the framework of PB model.

We take the boundary integral PB model as the example, other finite difference or finite element PB solvers will have the similar configuration.

Consider the discretized forms of Eqs. (3.3a) and (3.3b) for governing potential and its normal derivatives as

$$Sx = Bq \tag{3.17}$$

where $x \in \mathbb{R}^{2N}$ is the vector of potential and its normal derivative at the N collocation location, $q \in \mathbb{R}^{N_c}$ is the vector of N_c charges at atomic centers, $S \in \mathbb{R}^{2N\times 2N}$ and $B \in \mathbb{R}^{2N\times N_c}$ are the corresponding matrices to make the discretized forms Eqs. (3.3a) and (3.3b) valid. Similarly, the discretized form of part of Eq. (3.11) to find potential inside the protein [53] is

$$\phi = \phi_{\text{reac}} + \phi_{\text{coul}} = Rx + Dq \tag{3.18}$$

for $R \in \mathbb{R}^{N_c\times 2N}$, $D \in \mathbb{R}^{N_c\times N_c}$ and the discretized forms of Eqs (2.26) and (3.11) are

$$E_{\text{free}} = \frac{1}{2}q^T\phi = \frac{1}{2}q^T(RS^{-1}B + D)q = \frac{1}{2}q^TWq \tag{3.19}$$

and

$$E_{\text{solv}} = \frac{1}{2}q^T\phi_{\text{reac}} = \frac{1}{2}q^T RS^{-1}Bq = \frac{1}{2}q^T Aq \tag{3.20}$$

where

$$W := RS^{-1}B + D \tag{3.21}$$

and

$$A := RS^{-1}B \tag{3.22}$$

.

These simple and neat linear algebra equations give us an important clue to see q as the collection of all charges (location and charge quantity) and ϕ_{reac} at the charge location are the two most critical quantities in characterizing the protein structure, the force field, and the protein properties in the Poisson–Boltzmann model. Based on this assumption, the electrostatic features should be extracted from these two quantities. All our efforts in developing accurate and efficient PB model can be carried by ϕ_{reac} at the charge location.

The algorithm for obtaining the mulit-scale, physics-informed, uniform electrostatic features is explained in Figure 3.4. In short, the electrostatic of the protein from q and ϕ_{reac} at the charge locations will be represented by the point-multipoles, whose moments will be calculated using the treecode algorithm. To understand the point-multipoles, we provide some explanations below.

Consider a protein with N_c atoms and its multi-scale N_d point multipole representation (number of clusters for L levels in the tree structure) as shown in Fig. 3.4 for our machine learning model. Our goal is to compute the permanent multipole

$$\mathbf{M} = [\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^{N_d}]^T \tag{3.23}$$

as the uniform and scale-free feature for input to the machine learning model. Each term in $\mathbf{M}$ is the pth order expansion about the cluster center. For $i = 1, \ldots, N_d$

$$\mathbf{M^i} = [q^i] \text{ for } p = 0$$
$$\mathbf{M^i} = [q^i, d_x^i, d_y^i, d_z^i] \text{ for } p = 1$$
$$\mathbf{M^i} = [q^i, d_x^i, d_y^i, d_z^i, Q_{xx}^i, Q_{xy}^i, \ldots, Q_{zz}^i] \text{ for } p = 2$$

where q, d_i, Q_{ij} for $i, j = 1, 2, 3$ are the moments of the monopole, dipole, quadruple in suffix notation. In more detail, consider $n = 1, \cdots, N_c$, atoms at the nth cluster, centered at $\mathbf{r}_n = (x_n, y_n, z_n)$, the nth permanent order 2 (for example) multipole $\mathbf{M}^n$ consists of 10 components (excluding symmetric terms): $\mathbf{M}^n = [q^n, d_x^n, d_y^n, d_z^n, Q_{xx}^n, Q_{xy}^n, \ldots, Q_{zz}^n]^T$. Using this notation, the permanent charge at $\mathbf{r}_n$ can be written as [62, 63]

$$\rho^n(\mathbf{r}) = q^n \delta(\mathbf{r} - \mathbf{r}_n) + d_i^n \partial_i \delta(\mathbf{r} - \mathbf{r}_n) + Q_{ij}^n \partial_{ij} \delta(\mathbf{r} - \mathbf{r}_n), \tag{3.24}$$

A key idea is that the Coulomb potential G^n governed by the Gauss's law $-\Delta G^n = 4\pi\rho^n$ in the free space is expressed in terms of the *Green's function*

$$G^n(\mathbf{r}) = \frac{1}{|\mathbf{r} - \mathbf{r}_n|} q^n + \frac{r_i - r_{n,i}}{|\mathbf{r} - \mathbf{r}_n|^3} d_i^n + \frac{(r_i - r_{n,i})(r_j - r_{n,j})}{2|\mathbf{r} - \mathbf{r}_n|^5} Q_{ij}^n. \tag{3.25}$$

For all permanent multipoles $\mathbf{M} = [\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^{N_d}]^T$, the total Coulomb potential is *additive* such as $G^{\mathbf{M}}(\mathbf{r}) = \sum_{n=1}^{N_d} G^n(\mathbf{r})$ by the superposition principle.

For the point-multipole approach in the left picture of Fig 3.4, our goal is thus the computation of $\mathbf{M}$ accurately and efficiently. In fact, the computational cost is $O(N_c)$ using the strategies in [64], i.e. the moments at the finest cluster are computed first and a M2M (moments to moments) transformation can be used to efficiently compute moments at any desired level. These moments are intrinsic properties of the cluster thus can serve as features

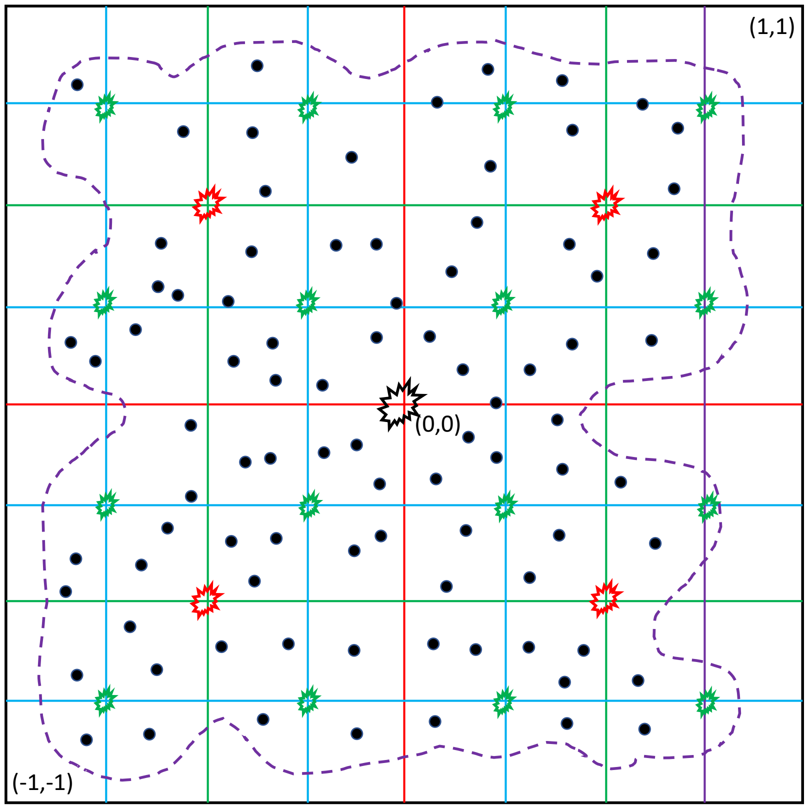


Figure 3.4: A 2-d illustration for 3-d uniform and multi-scale electrostatic features for a protein (purple dashed line) with charges (shown as black dots); charges q and reaction potential ϕ_{reac} are redistributed as point-multipoles (shown as explosion symbols) using Cartesian treecode [52] or FMM [64] at the centers of the cluster at different levels (level 0: black; level 1: red; level 2: green)

for the protein, which carries simplified and important information. The number of features up to level L cluster and pth order is

$$N_f(p, L) = N_p N_d \tag{3.26}$$

where $N_p = 1, 4, 10, 20, 35, 56, ...$ and $N_d = (1 + 8 + 8^2 + \cdots + 8^L) = \frac{8^{L+1}-1}{7}$.

3.3.1. Implementation Details

We outline the steps to compute the permanent multipole $\mathbf{M}$.

1. Read in the protein or protein-ligand complex which is represented by its atomic locations and partial charges: $q_i(\mathbf{r}_i)$ for $i = 1, \cdots, N_c$, e.g. the `pqr` file.
2. Calculate $N_f(p, L)$ based on user inputted parameters. These $N_f(p, L)$ numbers are ordered by level from 0 to L and the coordinates of cluster centers in each level. This determines the dimension of our final feature vector $F \in \mathbb{R}^{N_f}$.
3. Build the tree with L levels.
4. Compute the terms in the pth order Taylor expansion at each center and fill F.

Table 3.3 shows how N_f varies with the change of p and L.

3.4. Topological Data Analysis

Here we describe the theory of how we compute topological features which will be used later as inputs to our machine learning models for the prediction of binding affinity, Coulomb energy and solvation energy (computed using both Poisson-Boltzmann and Generalized Born methods). This strategy is based on a known method, called TopologyNet [65], which introduces the element-specific persistent homology (ESPH) method. This method abstracts

Table 3.3: Total number of features based on p and L

$p \backslash L$	0	1	2	3	4
0	1	9	73	585	4681
1	4	36	292	2340	18724
2	10	90	730	5850	46810
3	20	180	1460	11700	93620
4	35	315	2555	20475	163835

the information in the geometrically complex 3D biomolecular data, which represents the proteins, to create an image-like representation to take advantage of a very powerful tool in machine learning: convolutional neural networks. Utilizing these abstract images rather than the raw data directly overcomes computational challenges due to the size of large datasets of large proteins. The motivation behind the persistent homology idea is to provide a high level of abstraction of biological information. This is a classic example of a dimensionality reduction problem: the goal is to represent high-dimensional data with a low-dimensional representation. The main idea is to extract topological invariants, which are "the intrinsic features of the underlying space, of a given data set without additional structure information" [65].

First, define a simplicial complex as a topological space that is constructed from geometric components of a data set (or a point cloud), including discrete vertices and edges. Here, the vertices represent atoms in a biomolecule and the edges represent bonds in the biomolecule.

- A 0-simplex is a vertex
- A 1-simplex is an edge
- A 2-simplex is a triangle
- A 3-simplex is a tetrahedron

A linear combination of $k-$simplexes is called a $k-$chain. There are various ways to identify connectivity by varying a filtration parameter such as the radii of balls centered at the nodes. We can then generate complexes from the data, the two main types, following different rules, are a Cech or Rips complex:

- The k-simplicies of the **Cech complex** are determined by unordered $(k+1)$-tuples of points whose $\epsilon_t/2$ ball neighborhoods have a point of common intersection.
- The k-simplices of the **Rips complex** are determined by unordered $(k+1)$-tuples of points that are pairwise within distance ϵ_t, as seen in Figure **??**, where ϵ denotes the proximity parameter [66].

Identifying the *optimal* value of ϵ_t is a difficult task because certain values of ϵ_t may capture certain topological features of the data while others may not. For example, for sufficiently small values of ϵ_t, the complex is a discrete set, while for large values of ϵ_t, the complex is one high dimensional simplex. From this, we can define a persistent homology, which is constructed via a filtration process (a nested sequence of subcomplexes), in which the connectivity of the given dataset is systematically reset according to a scale parameter. Then, persistent homologies are visually represented via barcodes. A barcode is a graphical representation of a homology with "horizontal line segments in a plane whose horizontal axis corresponds to the parameter and whose vertical axis represents an (arbitrary) ordering of homology generators" [66]. Barcodes represent the persistence of a dataset's topological features over different spatial scales. We display examples of barcodes from our protein dataset later in Chapter 4. Betti-0 is the number of connected components, Betti-1 is the number of tunnels or circles and Betti-2 is the number of cavities or voids.

This persistent homology method holds promise, but it can oversimplify biological information, which is what motivated authors in [65] to extend the idea further to *element specific* persistent homology. This approach is able to characterize the biomolecular structures in

	Point	**Circle**	**Sphere**	**Torus**
Betti-0	1	1	1	1
Betti-1	0	1	0	2
Betti-2	0	0	1	1

Table 3.4: Betti numbers for different topological shapes

terms of multichannel topological invariants and considers specific atom types which may be associated with various interaction networks. Commonly occuring atoms for proteins include C, N, O and S and for ligands includes C, N, O, S, P, F, Cl, Br and I. They construct to element specific topological finger prints by including one type of atom from the protein and ligand, results in 36 cases.

Now to utilize the barcodes in a convolutional neural network framework a few pre-processing steps must occur. The barcodes are transformed into 1D image-like inputs by generating feature vectors which are defined as

$$V_i^b = \left\| \left\{ (b_j, d_j) \in \mathbb{B}(\alpha, \mathcal{C}, \mathcal{D}) \mid \frac{(i-1)L}{n} \leq b_j \leq \frac{iL}{n} \right\} \right\|, \quad 1 \leq i \leq n \tag{3.27}$$

$$V_i^d = \left\| \left\{ (b_j, d_j) \in \mathbb{B}(\alpha, \mathcal{C}, \mathcal{D}) \mid \frac{(i-1)L}{n} \leq d_j \leq \frac{iL}{n} \right\} \right\|, \quad 1 \leq i \leq n \tag{3.28}$$

$$V_i^p = \left\| \left\{ (b_j, d_j) \in \mathbb{B}(\alpha, \mathcal{C}, \mathcal{D}) \mid \frac{(i-1)L}{n} \geq b_j, \frac{iL}{n} \leq d_j \right\} \right\|, \quad 1 \leq i \leq n, \tag{3.29}$$

where b_j and d_j refer to the birth and death of certain topological features as ϵ_t increases, respectively. $\mathbb{B}(\alpha, \mathcal{C}, \mathcal{D})$ represents the collection of barcodes with α labeling the section of atoms depending on atom types and whether or not the atom is from the protein or ligand, $\mathcal{C}$ is the type of simplicial complex, and $\mathcal{D}$ indicated the dimension such as Betti-0, Betti-1 or Betti-2. The filtration interval is $[0, L]$ and is divided into n equal length subintervals, which can each be thought of as a pixel and $\frac{L}{n}$ can be viewed as the resolution. Now each channel depends on the specific atoms chosen. For instance, we adopt the methods established in [65] to end up with images of size 200 by 72. The filtration interval is taken to be $[0, 50]$ with bins of length 0.25. We have 72 channels from 36 due to the cases previously mentioned for V^d, then V^b, V^d and V^p for Betti-1 and Betti-2 barcodes for carbon atoms and all heavy atoms from the protein and protein-ligand complex counts 24 more. Then, the difference between the characterization of the protein and complex contributes 12 more cases. We see examples of these images in Chapter 4. These feature vectors can be thought of as one-dimensional images and thus we use *Conv1D* layers in our convolutional network architecture, as described below.

3.5. Machine Learning Models

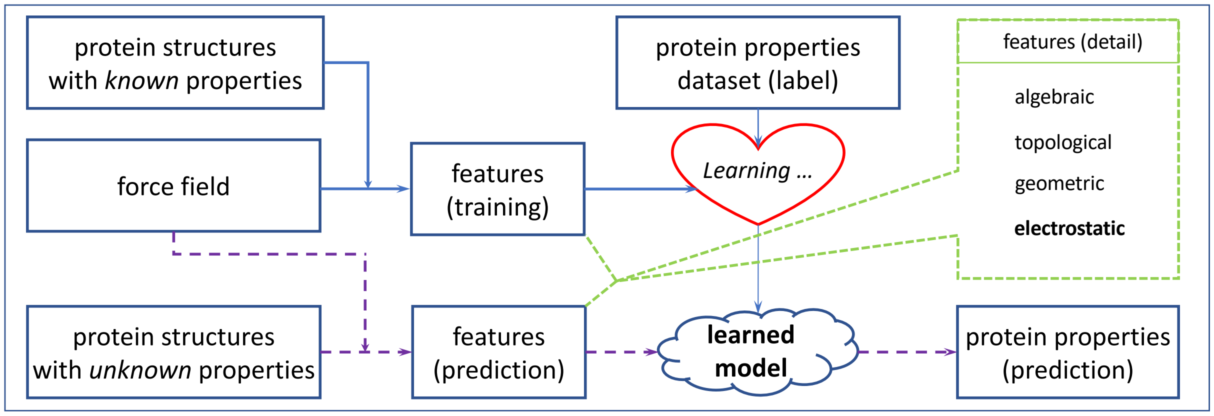

protein structures with *known* properties
force field
features (training)
protein properties dataset (label)
Learning ...
features (detail)
algebraic
topological
geometric
electrostatic
protein structures with *unknown* properties
features (prediction)
learned model
protein properties (prediction)

Figure 3.5: The DNN based Machine Learning model which uses protein structural data, force field, and known protein properties to mathematically generate algebraic, geometric, topological, and electrostatic features to train a learned model and then use the learned model to predict unknown properties for protein with available structures.

The main goal of this project is to discover the hidden and useful information embedded in the protein structural data from the protein data bank in a simple and abstract way. The protein structure data and force field include bonded and non-bonded interactions with which the molecular simulation can be performed. We categorize these information into *algebraic*, *geometric*, *topological* and *electrostatic* features. These four aspects are all important to the machine learning model.

The DNN-based machine learning model is shown in Fig. 3.5, which uses the available protein structural data [3], force field such as AMBER [67], CHARMM [68], AMOEBA [69], etc. and known protein properties repositories such as PDBbind Database [70], Protein pKa Database [71], etc to mathematically generate *algebraic*, *geometric*, *topological*, and *electrostatic* features to train a learned model and then use the learned model to predict unknown properties for protein with available structures. Here, we focus on the topological and electrostatic features. Details on algebraic and geometric feature development/extraction techniques can be found in [61].

3.5.1. Network Architectures

The first model we explore is for the prediction of solvation energy (calculated with the PB model and GB model) prediction using the scale-free uniform electrostatic features described above. We first preprocess the data by removing outliers using the interquartile range (IQR) method. The features, represented by X are standardized with mean 0 and standard deviation of 1 using *StandardScaler()* from *sklearn.preprocessing*. Having the features on the same scale helps the neural network converge for this regression task. We also choose to standardize the output labels, represented by y because of the wide range of the solvation energy values. We make sure to invert the transform from the predicted output for our reported scatter plots in Chapter 4.

We use a keras Sequential model with each layer outlined in Figure 3.6. The convolutional neural network starts with 1D convolution layers, followed by fully connected (Dense) layers. This is advantageous because the convolutional network learns higher-level features from the earlier layers in the model then once the output is flattened more dense layers follow to ultimately perform regression. The input data to the model has $N(p, L)$ features. Notations are Dense(neuron number, activation function, weight initialization, regularization constant), Dropout(dropout rate) and Activation(activation function). We use batch normalization to normalize the activations of a previous layer at each batch and dropout to randomly deactivate neurons. Both of these approaches help to reduce overfitting.

We use 500 epochs to predict solvation energy (calculated from both the PB model and GB model as well as Coulomb energy. We use the adaptive momentum (adam) optimizer with a learning rate of 0.0001 and batch size of 32. We use mean squared error as our loss function. Figure 3.7 outlines the model architecture used for the convolutional neural network. Notations are Conv1D(filter number, filter size, activation function, regularization constant) and AveragePooling(pooling size). The output layer for the regression task uses a linear activation function. The combined model in Figure 3.8 uses both electrostatic and topological features. The topological features are inputted into the convolutional layers on the left side of the figures while the electrostatic features are inputted into the model with Dense layers, as seen on the right side of Figure 3.8. The outputs of these models are then merged and used as the input to the final layers consisting of more dense layers with the final output being the solvation energy calculation from the PB model. Hyperparameters were chosen based on trial and error and also motivated by the hyperparameters chosen in [65] which utilized a parameter search using Hyperopt [72].

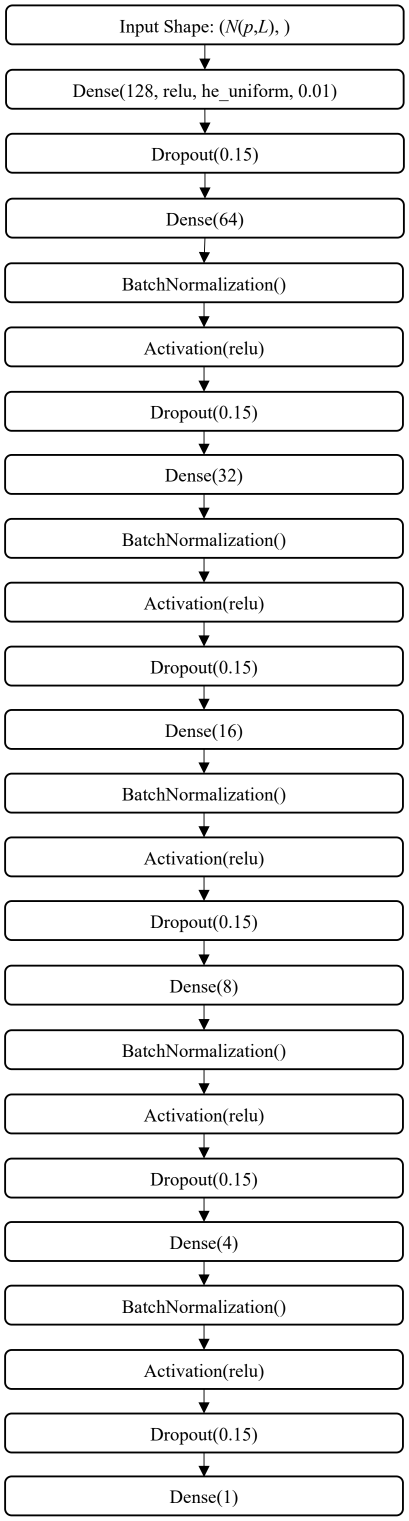


Figure 3.6: Network architecture using solely electrostatic features to predict 1) solvation energy calculated by PB model, 2) solvation energy calculated by GB model and 3) Coulomb energy

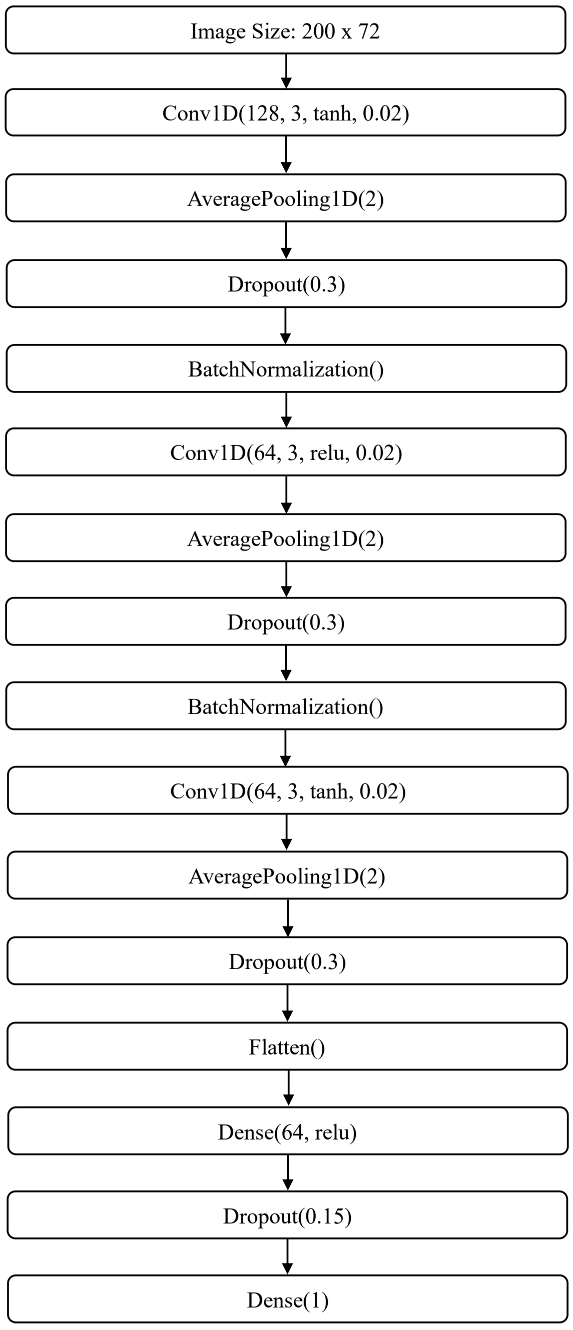


Figure 3.7: Network architecture using solely topological features to predict solvation energy calculated by the PB model

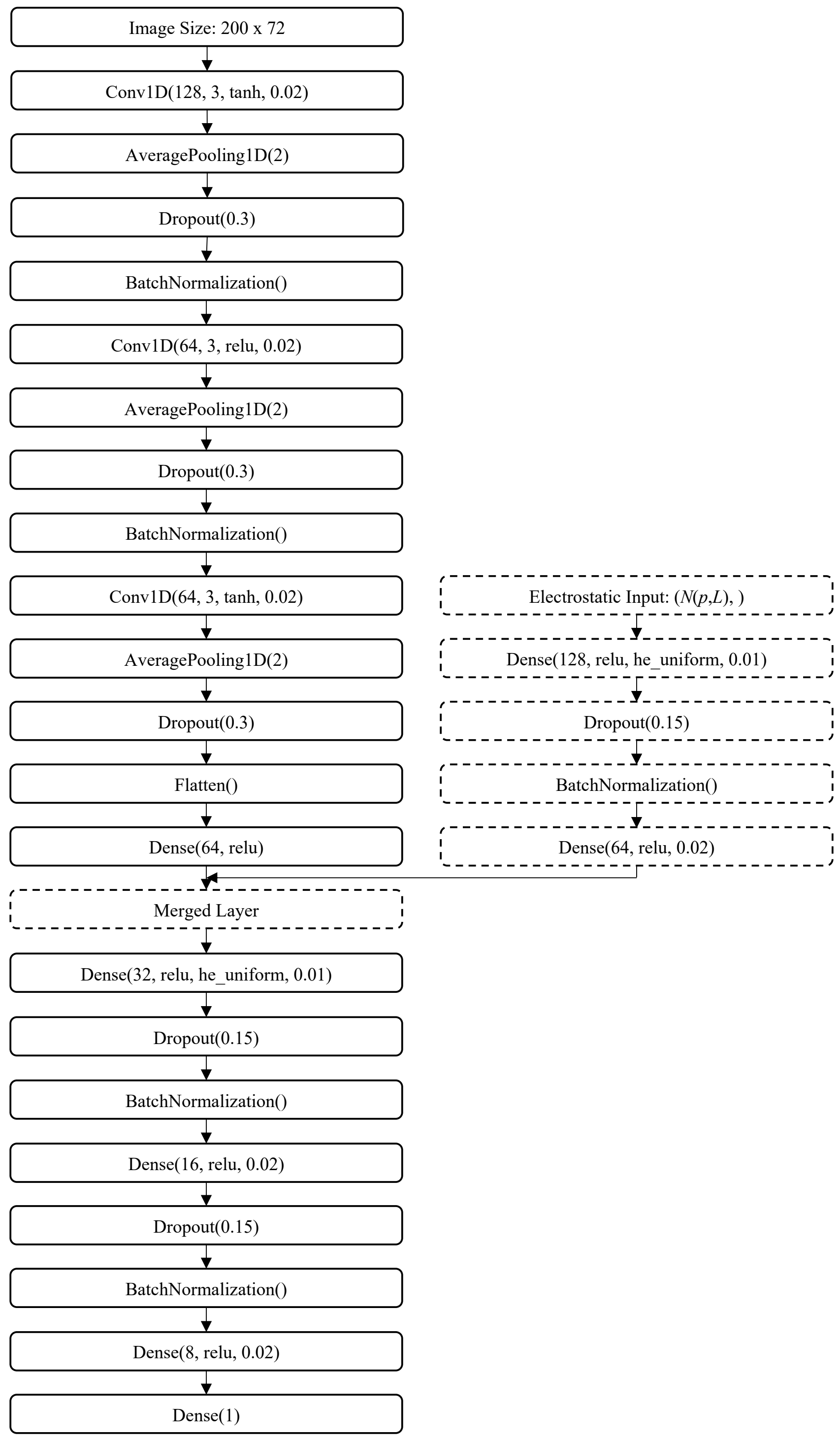


Figure 3.8: Network architecture using both electrostatic and topological features to predict solvation energy calculated by the PB model

Chapter 4

Results

This chapter contains all the numerical simulation results, ranging from the electrostatics on COVID-19 protein results using the parallel boundary integral PB solvers, to the performance of our DNN model using topological and electrostatic features.

4.1. Performance of Parallelized Poisson-Boltzmann Solvers

This section focus on the parallelization performance of the boundary integral PB solvers. We first demonstrate the load balance optimization using the designed cyclic schemes, followed by a comparison study between the MPI-based TABI solver and the GPU-accelerated DSBI solver for the determination of a size-threshold to use the more suitable solver. An accurate, efficient, and reliable PB solver makes it possible for us to generate labels i.e. the electrostatic solvation energies of the collection of proteins for the machine learning model, whose performance is reported at the end of this chapter.

4.1.1. Sequential vs Cyclic

Our numerical results are generated on supercomputers sponsored by Southern Methodist University's Center for Research Computing (CRC). The MPI-based results are generated on M3 (https://www.smu.edu/oit/services/m3) and GPU-based results are generated on SuperPOD (https://www.smu.edu/oit/services/superpod).

We first check the parallel efficiency of our MPI-based algorithm with both sequential and cyclic schemes by computing the solvation energy on protein 7n3c at MSMS density of

12, which generates 529,911 boundary elements. We use up to 256 MPI tasks and Table 4.1 shows the results. Column 1 shows the increasing numbers of MPI tasks. Column 2 reports the Total CPU time when the direct sum (DS)BI scheme is used to compute the electrostatic solvation free energy. Due to its $O(N^2)$ computational cost, the CPU time for the DSBI solver is overly long, even when 256 tasks are used. Columns 4 and 5 display the total CPU time and parallel efficiency for the TABI solver using the sequential and cyclic schemes, both of which are much faster compared to DSBI. Columns 8 and 9 focus more closely on the time required for a single matrix-vector product Ax, $\bar{t}_{Ax}$, which we take as the average of the iteration's maximum CPU time among all tasks,

$$\bar{t}_{Ax} = \frac{1}{n_i} \sum_{k=1}^{n_i} \max_j t_{Ax}^{j,k} \tag{4.1}$$

where $t_{Ax}^{j,k}$ is the CPU time to compute Ax from the jth task in the kth GMRES iteration.

Parallel efficiencies are displayed in Columns 3, 6, 7, 10 and 11. The parallelization of the DSBI solver shows high efficiency as seen in column 3. This is due to the simplicity of the algorithm. Other than the four stages identified in Figure 3.2, there is very little serial computation or communication required. However, the parallel efficiency of the TABI solver is not as good as the DSBI solver, as shown in Columns 6 and 7. This is primarily due to the use of treecode, which has some serial time for building the tree and computing the moments. The serial time is relatively short when n_p is small but becomes increasingly significant as n_p grows and the time spent within parallelized stages decreases. If we focus specifically on the parallelization of the treecode in computing Ax, Columns 10 and 11 show a high degree of parallel efficiency. We can also observe that the cyclic scheme significantly improves the parallel efficiency in comparison with the sequential scheme. However, due to the very small fraction of runtime spent in computing matrix-vector products as n_p increases, the overall parallel efficiencies from Columns 6 and 7 do not show a significant difference between the sequential and cyclic schemes. To more carefully examine the performance

Table 4.1: CPU time and parallel efficiency (P.E.) for parallelized direct sum, sequentially parallelized treecode (seq.) and cyclically parallelized treecode (cyc.) for computing electrostatic solvation energy (-6020.52 kcal/mol) for protein 7n3c with 529,911 boundary elements. The treecode parameters are $\theta = 0.8$, $N_0 = 100$, and $p = 3$; The number of tasks n_p ranges over $1, \dots, 256$. The time for one Ax ($\bar{t}_{Ax}$) is the average iteration's maximum CPU time over all tasks.

n_p	DSBI Solver		TABI solver							
	Total Time		Total Time				Time for one Ax ($\bar{t}_{Ax}$,)			
	CPU (s)	P.E.	CPU (s)		P.E.		CPU (s)		P.E.	
			seq.	cyc.	seq.	cyc.	seq.	cyc.	seq.	cyc.
1	106063.17	100.00	1874.88	1873.60	100.00	100.00	89.75	89.60	100.00	100.00
2	53132.86	99.81	971.25	967.12	96.52	96.87	45.49	45.22	98.63	99.07
4	26549.87	99.87	561.25	502.60	83.51	93.20	25.69	22.57	87.34	99.26
8	13291.47	99.75	321.42	285.25	72.91	82.10	13.94	12.02	80.46	93.22
16	6710.06	98.79	171.41	158.04	68.36	74.09	6.43	5.77	87.30	97.08
32	3928.71	84.37	128.13	114.73	45.73	51.03	3.99	3.26	70.22	85.84
64	2022.84	81.93	99.75	90.81	29.37	32.24	2.14	1.66	65.55	84.24
128	1042.49	79.48	79.82	76.83	18.35	19.05	1.08	0.85	64.65	82.77
256	554.50	74.72	71.70	71.16	10.21	10.28	0.56	0.45	62.27	78.61

differences between the sequential and cyclic decomposition schemes, in Fig. 4.1 we plot $\bar{t}_{Ax}$ from Eqn. (4.1) when 8,16,$\cdots$, 256 MPI tasks are used. It is evident that the cyclic scheme has reduced variance compared with the sequential scheme owing to its advantage in load balance.

4.1.2. MPI-based TABI Solver vs GPU-accelerated DSBI Solver

Next, we compute the solvation energy for the six COVID-19 proteins introduced previously using MSMS with density equal to 12 to provide sufficient detail of the molecular surface. We use both the MPI-based TABI solver and the GPU-accelerated DSBI solver. For a reasonable computing power comparison, we use 64 CPU cores for the MPI-related computing and 1 GPU card for the GPU-related computing. Table 4.2 shows the simulation results. Column 1 is the PDB ID for proteins in ascending sequence of their size followed by the number of atoms in column 2, number of boundary elements in column 3, and the areas of the solvent excluded surface in column 4. Columns 5 and 6 are the number of GMRES iterations, from which we can see that the TABI solver has much improved condition number in comparison with the DSBI solver thanks to the TABI preconditioner from Section 3.1.4. The solvation energies are reported in columns 7 and 8, which are sufficiently close. The differences are caused by the treecode approximation, the preconditioning scheme, and the error tolerance achieved when the iteration is stopped. Note for calculating protein electrostatic solvation energy, we don't have an exact value to compare. If the DSBI solver converges before reaching the maximum number of allowed GMRES iterations, its result should be more accurate than that from TABI solver since Treecode and Preconditioner could add extra approximations. For example, the $E_{\text{sol}}^{\text{GPU}}$ results from proteins 6yi3, 7act, 7n3c, 6wji should be more accurate than the $E_{\text{sol}}^{\text{MPI}}$ results for these proteins. However, if the GMRES allowed maximum number of iteration has been reached for examples for proteins 7cr5 and 7sts, on which the DSBI solver stopped when 100 iterations are reached while the accuracy did not

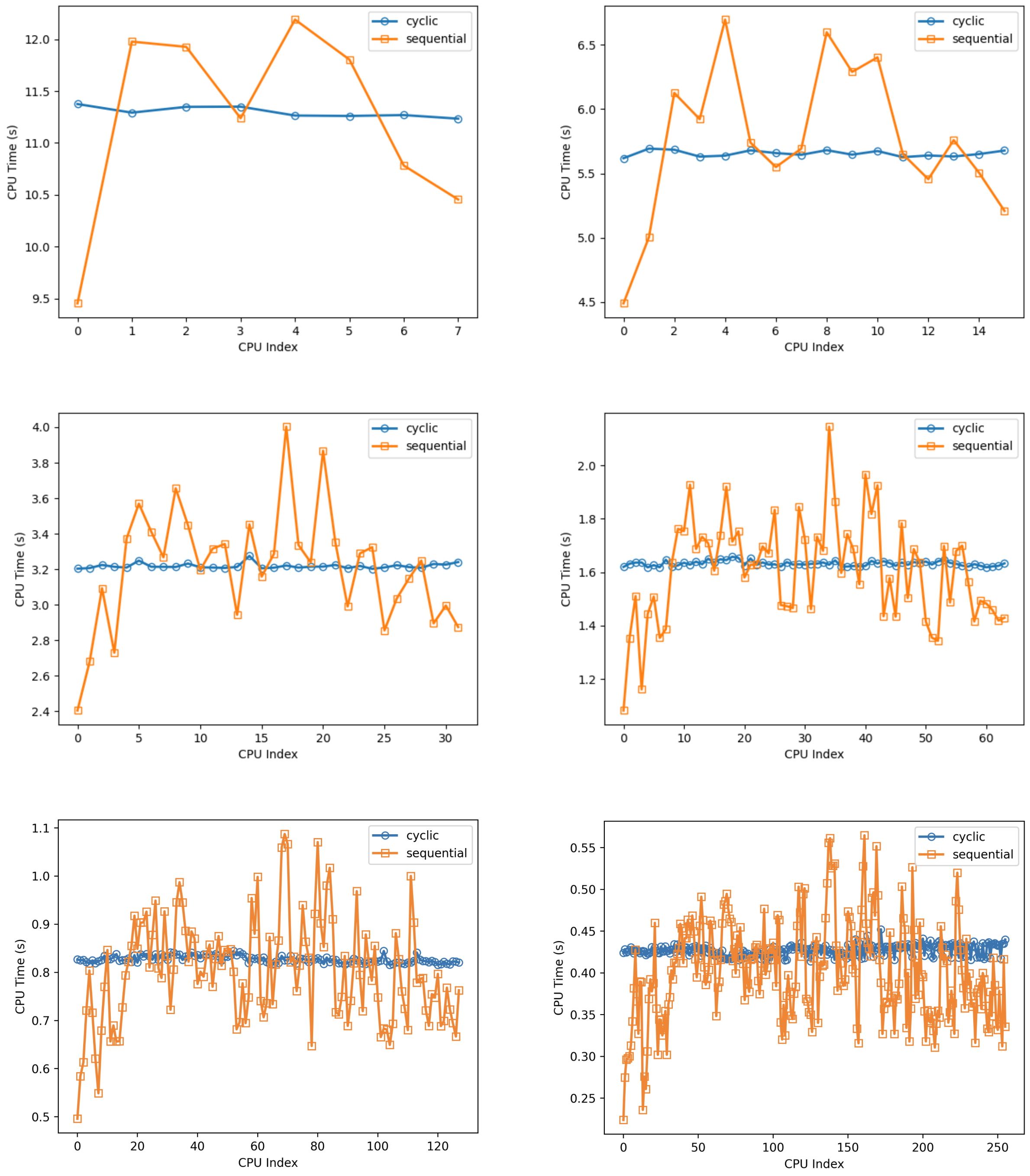


Figure 4.1: MPI-based parallelization with sequential and cyclic schemes for 8, 16, $\cdots$, 256 tasks. The CPU time reported is $\bar{t}_{Ax}$, the average GMRES iteration's maximum CPU times among all tasks.

meet the 10^{-4} threshold, we can not say for sure whether $E_{\mathrm{sol}}^{\mathrm{GPU}}$ result or $E_{\mathrm{sol}}^{\mathrm{MPI}}$ result is more accurate. The computation times are shown in columns 9 and 10, which demonstrates that the computing power between 64 CPUs and 1 GPU are comparable. However, the algorithms (with preconditioning vs without preconditioning, direct sum vs treecode) can make a substantial difference for ill-conditioned or larger systems. For example, for protein 7sts, with nearly one million boundary elements, the MPI-based TABI solver is significantly faster than the GPU-based DSBI because of the ill-conditioned system and the large size of the problem.

Table 4.2: Computing electrostatic solvation energies in (kcal/mol) for the involved proteins: ionic strength = 0.15M; $\epsilon_1 = 1$, $\epsilon_2 = 80$; MSMS [73] density=12; N_c is the number of atoms/charges, N is the number of boundary elements, n_i is the number of GMRES iterations, S_{ses} is the solvent excluded surface area, and E_{sol} is the electrostatic solvation energy.

PDB	N_c	N	S_{SES}	n_i^{MPI}	n_i^{GPU}	$E_{\mathrm{sol}}^{\mathrm{MPI}}$	$E_{\mathrm{sol}}^{\mathrm{GPU}}$	t^{MPI} (s)	t^{GPU} (s)
6yi3	2083	169,968	7516.44	10	10	-1941.81	-1945.18	14.76	8.96
7act	2352	188,054	8286.70	14	14	-1893.88	-1934.49	21.35	17.39
7cr5	8133	513,226	22524.23	16	100+	-5713.52	-5786.69	89.52	695.17
7n3c	8459	530,084	23244.85	19	17	-6020.52	-6013.68	99.13	132.20
6wji	10182	641,266	28116.88	13	14	-14009.55	-14016.02	112.82	152.71
7sts	15797	993,572	43457.63	26	100+	-11622.63	-11583.26	422.67	2544.70

We then further investigate under what conditions we should choose between using the GPU-accelerated DSBI solver or MPI-based TABI solver. The following example, whose result is shown in Table 4.3, gives some important guidance. In this example, we compute the solvation energy for protein 6yi3 for increasing values of the MSMS density (d), giving rise to increased problem sizes, as shown in columns 1 and 2. Columns 3 and 4 show the similar solvation energy computed with these two approaches. Columns 5 and 6 report the number of GMRES iterations. From these close results, we see that the discretized system for this protein is well conditioned thus the preconditioning scheme has limited effect. We solve the problem using 1 CPU core and report the time in column 7 for reference. Then we report the time for solving the problem using 64 MPI tasks and one A100 GPU card in columns 8 and 9. The result indicates that for a protein whose discretized system is well-conditioned, when the number of boundary elements is less than 250,000, we should use the GPU-accelerated DSBI solver, since the smaller the system the better the GPU-accelerated DSBI solver compares against the MPI-based TABI solver. If the conditioning of A shows a pressing need for preconditioning, the threshold number will be smaller for the GPU-accelerated DSBI solver. The rapid GPU performance at least gives us the hope to perform molecular dynamics or Monte Carlo simulation for small and middle-sized proteins using GPUs. For example, if 50,000 boundary elements can reasonably describe the given protein, a single PB equation solution only takes about one second using one GPU card, in comparison with 4 seconds on a 64-core cluster.

Table 4.3: Computing electrostatic solvation energies in (kcal/mol) for the protein 6yi3 at different MSMS densities: ionic strength = 0.15M; $\epsilon_1 = 1$, $\epsilon_2 = 80$; d is the MSMS density, N is the number of boundary elements, n_i is the number of GMRES iterations, E_{sol} is the electrostatic solvation energy. Results are generated using KOKKOS and MPI on ManeFrame III; MPI results are from using 64 tasks; GPU results are from using one A100 GPU.

d	N	$E_{\text{sol}}^{\text{MPI}}$	$E_{\text{sol}}^{\text{GPU}}$	n_i^{CPU}	n_i^{GPU}	t_{CPU} (s)	t_{MPI} (s)	t_{GPU} (s)
2	28,767	-2057.61	-2056.26	10	10	29.34	2.76	0.83
4	56,127	-1999.01	-1997.03	10	10	66.40	4.46	1.52
6	84,903	-1968.00	-1966.87	10	16	108.52	7.05	4.98
8	110,307	-1954.62	-1952.23	10	10	145.13	9.35	4.32
12	169,955	-1945.18	-1941.81	10	10	240.54	14.76	8.91
16	229,901	-1940.96	-1936.87	10	11	340.82	19.86	17.66
18	257,236	-1938.27	-1933.67	10	11	385.96	**21.69**	**23.73**
20	287,202	-1937.18	-1931.77	10	12	438.86	24.54	28.73
24	343,806	-1933.63	-1928.65	10	11	534.31	35.13	38.62
28	407,196	-1933.04	-1927.56	10	12	653.06	41.84	55.18
32.5	471,307	-1931.76	-1926.04	10	12	760.12	51.23	77.83
64	946,335	-1928.51	-1921.81	10	13	1701.06	145.65	311.07

We note that the solution to the boundary integral PB equation gives both the electrostatic potential and its normal derivative on the molecular surface. We can plot the potential on the surface elements. The color-coded potential can provide guidance on the docking site for the ligand, or offer other insights pertaining to protein-protein interactions. Some examples of this kind of visualization are shown in Fig. 4.2.

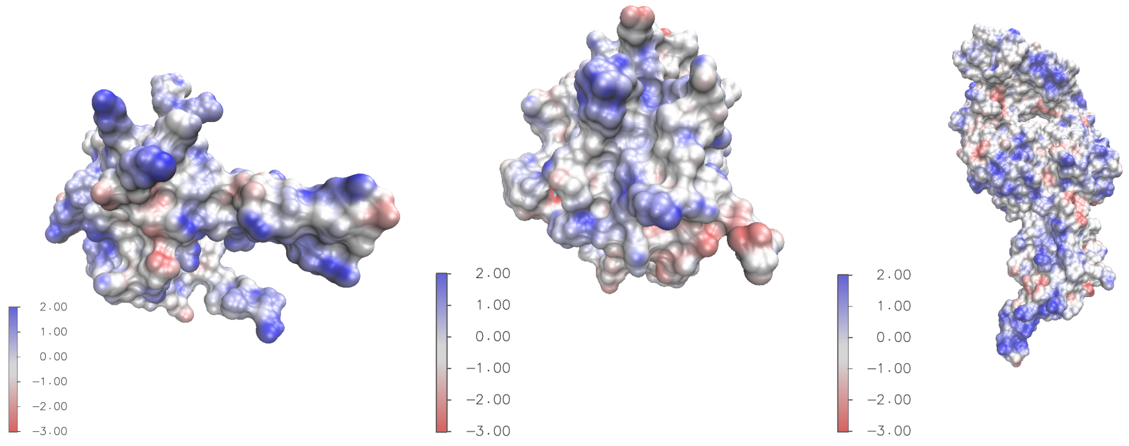


Figure 4.2: Color coded electrostatic surface potential in kcal/mol/e_c on the molecular surface of proteins 6yi3 (left), 7act (middle), and 7n3c (right); plot is drawn with VMD [74].

4.2. Performance of Machine Learning Models

Starting from this section, we report the performance of our machine learning models to predict the protein properties such as binding affinity, Coulomb energies, electrostatic solvation energies, etc. To differentiate the significance of topological and electrostatic features, we report the model performance using these features alone and together.

4.2.1. Binding Affinity Prediction Using Topological Features

Topological features play a significant role in our machine learning models as they abstract one-dimensional topological invariants from the complex 3-d protein structures. To this end, we adopt the schemes from TopologyNet [65]. We have generated topological features for the PDBBind 2007 core set. 195 protein-ligand complexes were used as the test set and the PDBBind 2007 refined set, excluding the PDBBind 2007 core set, was used as the training set which contained 1,105 protein-ligand complexes. The Gudhi package was used to generate topological features which are visualized in Figure 4.3 with their binding affinity. These topological features were then inputted into the convolutional neural network described in [65]. Model performance was measured in terms of the Pearson correlation coefficient between the true affinity values y_{true} and the model predicted affinity values y_{pred}. We achieved a Pearson correlation coefficient of approximately 0.78 using the model architecture in [65], which achieved a Pearson correlation coefficient of 0.82. Results for the TopologyNet binding predictor are displayed in the top portion of Figure 4.4. The loss on the left shows evidence of overfitting as the validation loss converges at a much higher value than the training loss. This suggests the model's inability to generalize to unseen data, such as the test set here. This motivated the design of a simpler model to reduce the effects of overfitting. The results for the modified model designed to reduce overfitting are displayed in the bottom part of Figure 4.4. The architecture for this model is outlined in Figure 3.7. The Pearson correlation coefficient for this model is 0.75. In [65], authors report the Pearson correlation coefficients of various scoring methods for the prediction of protein-ligand binding affinity. Our reported result ranks number 3 out of the 25 methods reported. Despite this, the uniform scale-free electrostatic features did not prove effective for the prediction of protein-ligand binding affinity. Therefore, the electrostatic features showed limited benefit when combined with the topological features for predicting binding affinity.

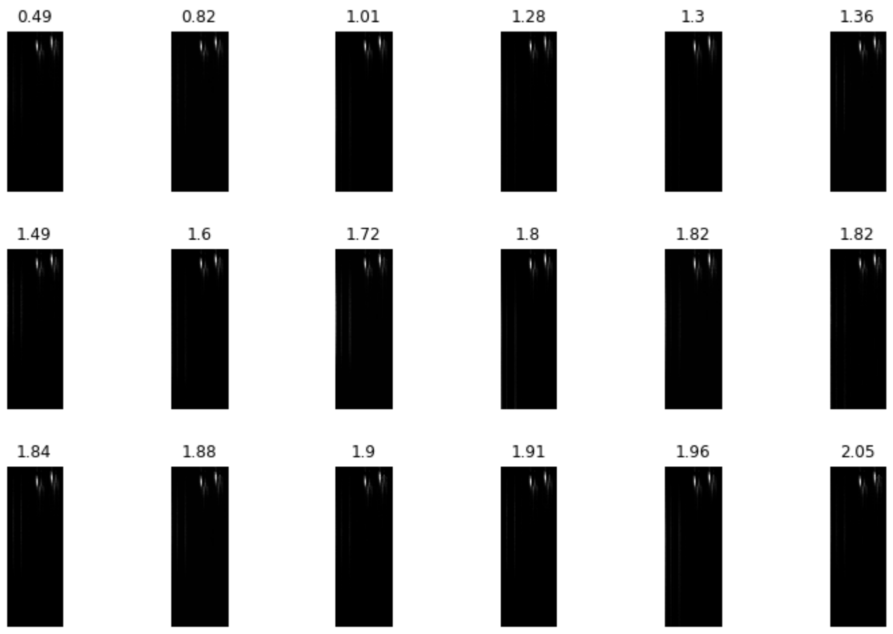


Figure 4.3: Topological features with corresponding binding affinity values

4.2.2. Solvation and Coulomb Energy Prediction Using Electrostatic Features

The weak contribution from electrostatic features in predicting binding affinity implies the weak connection between the binding affinity, which is from experiments, and protein electrostatics. In fact, a consultation from authors of [65] verified our doubts since they once tested the correlation between electrostatic binding energy and the experimental binding affinity, which is rather weak. Our attention then moved to the prediction of Coulomb energy and solvation energy of proteins using our machine learning models.

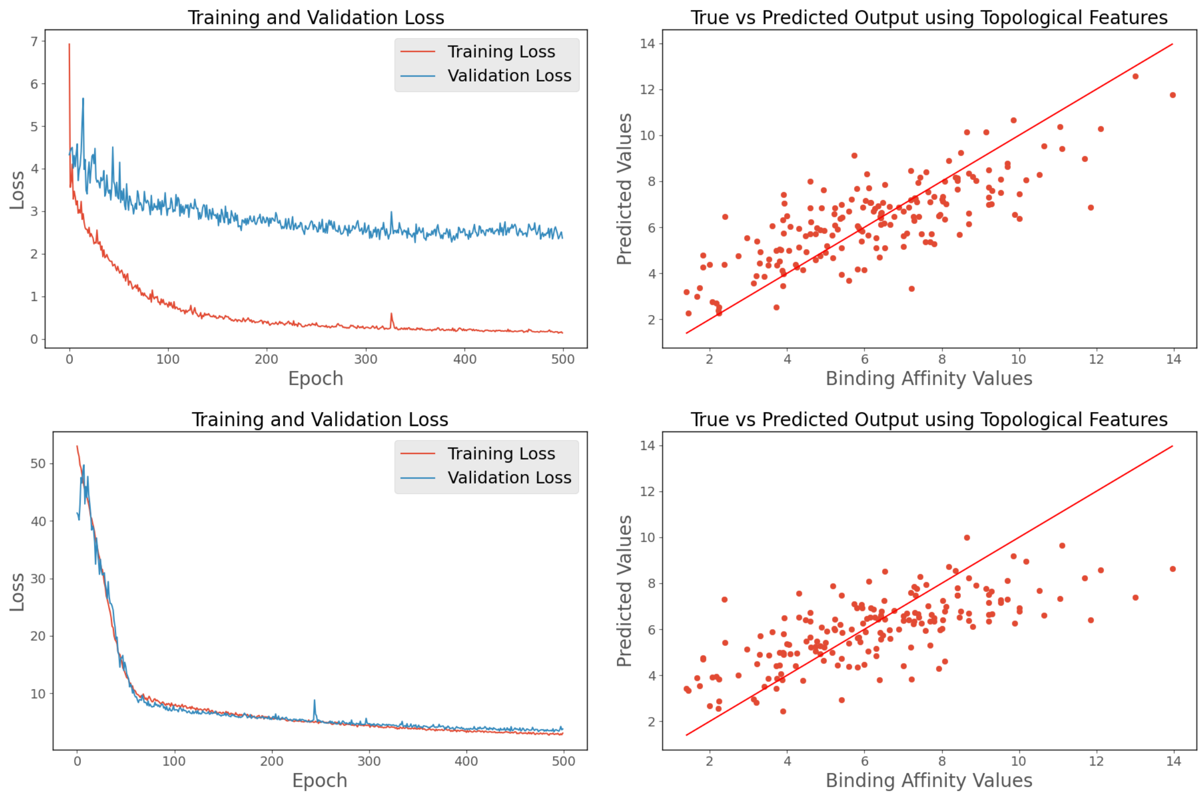


Figure 4.4: Modified model designed to reduce overfitting loss (left) and scatter plot of true vs. predicted affinity values of binding affinity

We used the v2018 dataset consisting of 4658 proteins. Removing outliers using the interquartile range method reduced this number to 4295 proteins and 80% of the data was used as the training set while 20% was reserved for the testing set. Further, we used 5-fold cross-validation to train the model. The evaluation metrics used are

1. MSE (Mean squared error) between the scaled y_{true} and scaled y_{pred}
2. R^2 (explained variance) between y_{true} and y_{pred}
3. Pearson correlation coefficient between y_{true} and y_{pred}
4. MAPE (Mean absolute percentage error) between y_{true} and y_{pred}

and are reported on the test set in the following tables. The following loss plots displayed in this chapter are the mean loss over $k = 5$ folds. It is important to note that the cost to compute these electrostatic features is low, making our approach more efficient computationally compared to solving the PB model in its entirety to compute the solvation energy. This relates to our initial goal to develop fast and efficient algorithms. The time to compute the features of 4658 proteins, with sizes shown in Figure 4.2.2, for various combinations of p and L for the v2018 dataset is reported in Table 4.4, which is very efficient!

Table 4.4: Time in seconds to compute electrostatic features for v2018 dataset consisting of 4658 proteins

p	L	$N(p, L)$	Time (s)
1	0	4	107.09
2	0	10	107.70
3	0	20	111.66
4	0	35	116.35
0	1	9	111.72
1	1	36	107.79
2	1	90	111.44
3	1	180	117.89
4	1	315	129.07
0	2	73	107.53
1	2	292	110.81
2	2	730	121.52
3	2	1460	138.47
4	2	2555	161.98
0	3	585	113.66
1	3	2340	130.95

To determine if these features are not only efficient to compute but also accurate in their ability to predict biologically relevant quantities such as Coulomb and solvation energy we examine the loss and scatter plots for the p and L cases in the next sections. We also examine if these electrostatic features, in combination with topological features, can improve model performance.

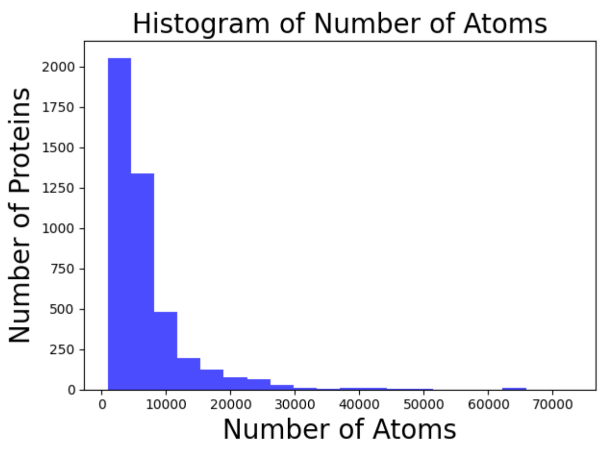


Figure 4.5: Histogram showing the number of atoms in each protein in the v2018 dataset

4.2.2.1. Coulomb Energy

The first prediction we make using our electrostatic features is Coulomb energy. Results of evaluation metrics are reported in Table 4.5. The first column is the order p, the second column is the level L, the third column $N(p, L)$ is the number of features generated from p and L, and the fourth through seventh columns are the evaluation metrics outlined above.

In Table 4.5, we see the optimal performance, in terms of the correlation coefficient, is $p = 4/L = 2$ with 0.92 as displayed in the fourth row of .Figure 4.6 We display the loss plots (left) and scatter plots (right) of y_{true} versus y_{pred} in Figure 4.6 for $p = 3/L = 1$, $p = 4/L = 1$, $p = 3/L = 2$ and lastly $p = 4/L = 2$. The loss plots demonstrate reasonable convergence and do not show obvious signs of overfitting, indicating that the scale-free uniform electrostatic features are relevant to the prediction of Coulomb energy. Notice that as we fix the number of levels in the tree L and increase the order p (as the table is organized) the MSE and MAPE generally decrease while the Pearson correlation coefficient and R^2 increase. This indicates that the model performance increases as the order p increases. This makes sense because as the order increases in the multipole expansion we get a more accurate approximation. This is advantageous as including more terms for higher accuracy does not come at a significantly higher cost in terms of computational time as seen in Table 4.4.

Table 4.5: Results for Coulomb energy prediction

p	L	$N(p,L)$	MSE	PCC	R^2	MAPE
1	0	4	0.76	0.61	0.34	0.33
2	0	10	0.44	0.81	0.62	0.25
3	0	20	0.40	0.85	0.66	0.23
4	0	35	0.31	0.88	0.73	0.21
0	1	9	0.86	0.54	0.25	0.34
1	1	36	0.40	0.83	0.65	0.23
2	1	90	0.25	0.89	0.79	0.18
3	1	180	0.23	0.90	0.80	0.16
4	1	315	0.24	0.90	0.79	0.17
0	2	73	0.52	0.75	0.55	0.26
1	2	292	0.34	0.89	0.70	0.20
2	2	730	0.33	0.90	0.71	0.17
3	2	1460	0.35	0.91	0.70	0.18
4	2	2555	0.27	0.92	0.77	0.17
0	3	585	0.57	0.75	0.50	0.24
1	3	2340	0.62	0.76	0.46	0.24

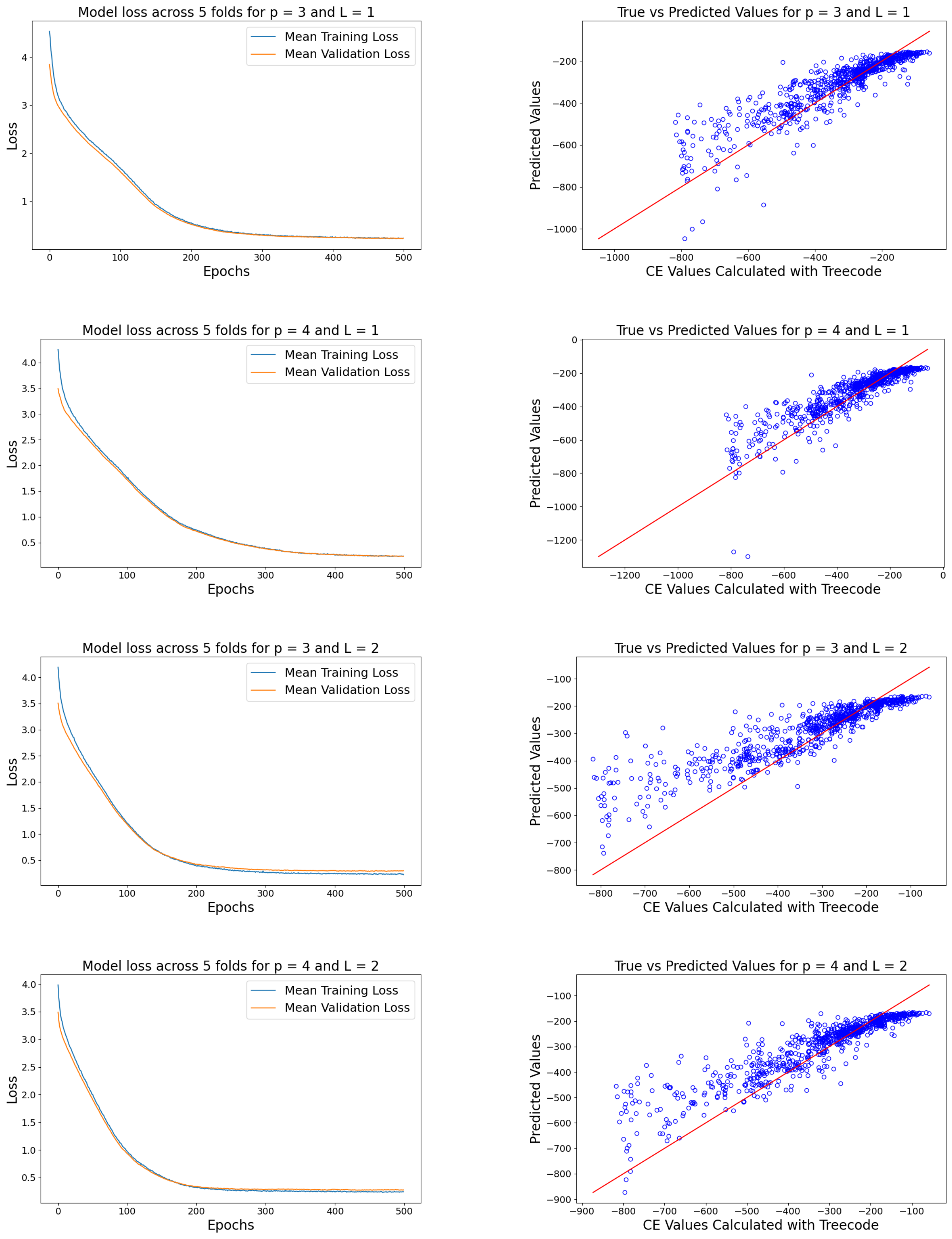


Figure 4.6: Loss plots (left) and scatter plots (right) for Coulomb energy prediction using electrostatic features for $p = 3/L = 1$, $p = 4/L = 1$, $p = 3/L = 2$ and $p = 4/L = 2$.

4.2.2.2. Solvation Energy From the Poisson-Boltzmann Model

Next, we investigate the prediction task of solvation energy computed from the Poisson-Boltzmann model using the aforementioned electrostatic features. In the previous sections, we see the time to compute the solvation energy for a single protein with the Poisson Boltzmann model motivated our design of parallelization using CPUs and GPU. Here, we know that we can quickly compute the electrostatic features, as seen in Table 4.4. We see improved performance on this task compared to predicting Coulomb energy, in terms of all evaluation metrics measured. More specifically, the best correlation coefficient is 0.95 for $p = 4/L = 0$ and then 0.94 for the cases of $p = 2/L = 1$, $p = 3/L = 1$ and $p = 4/L = 1$ which are plotted in Figure 4.7. We see a similar trend that as p increases, MSE and MAPE decrease while the correlation coefficient and R^2 increase as seen in Table 4.6. The loss plots on the left of Figure 4.7 show similar convergence for both the training and validation sets which suggests the model has learned the underlying patterns from the electrostatic features, without memorizing the training data. The scatter plots on the right column of 4.7 demonstrate a reasonable fit as many of the points follow the trend of the red line representing $y = x$. Overall, it seems as though the model is capturing the relationship between the electrostatic features and target variable, solvation energy computed by the Poisson-Boltzmann model.

Table 4.6: Results for solvation energy computed by Poisson-Boltzmann prediction

p	L	$N(p,L)$	MSE	PCC	R^2	MAPE
1	0	4	0.40	0.78	0.59	0.23
2	0	10	0.18	0.92	0.81	0.16
3	0	20	0.16	0.93	0.83	0.15
4	0	35	0.13	0.95	0.86	0.14
0	1	9	0.46	0.73	0.52	0.27
1	1	36	0.20	0.91	0.79	0.17
2	1	90	0.12	0.94	0.87	0.14
3	1	180	0.14	0.94	0.86	0.17
4	1	315	0.13	0.94	0.86	0.14
0	2	73	0.28	0.85	0.71	0.21
1	2	292	0.20	0.92	0.80	0.16
2	2	730	0.19	0.92	0.80	0.16
3	2	1460	0.19	0.92	0.80	0.16
4	2	2555	0.20	0.91	0.79	0.15
0	3	585	0.35	0.83	0.64	0.21
1	3	2340	0.45	0.84	0.54	0.21

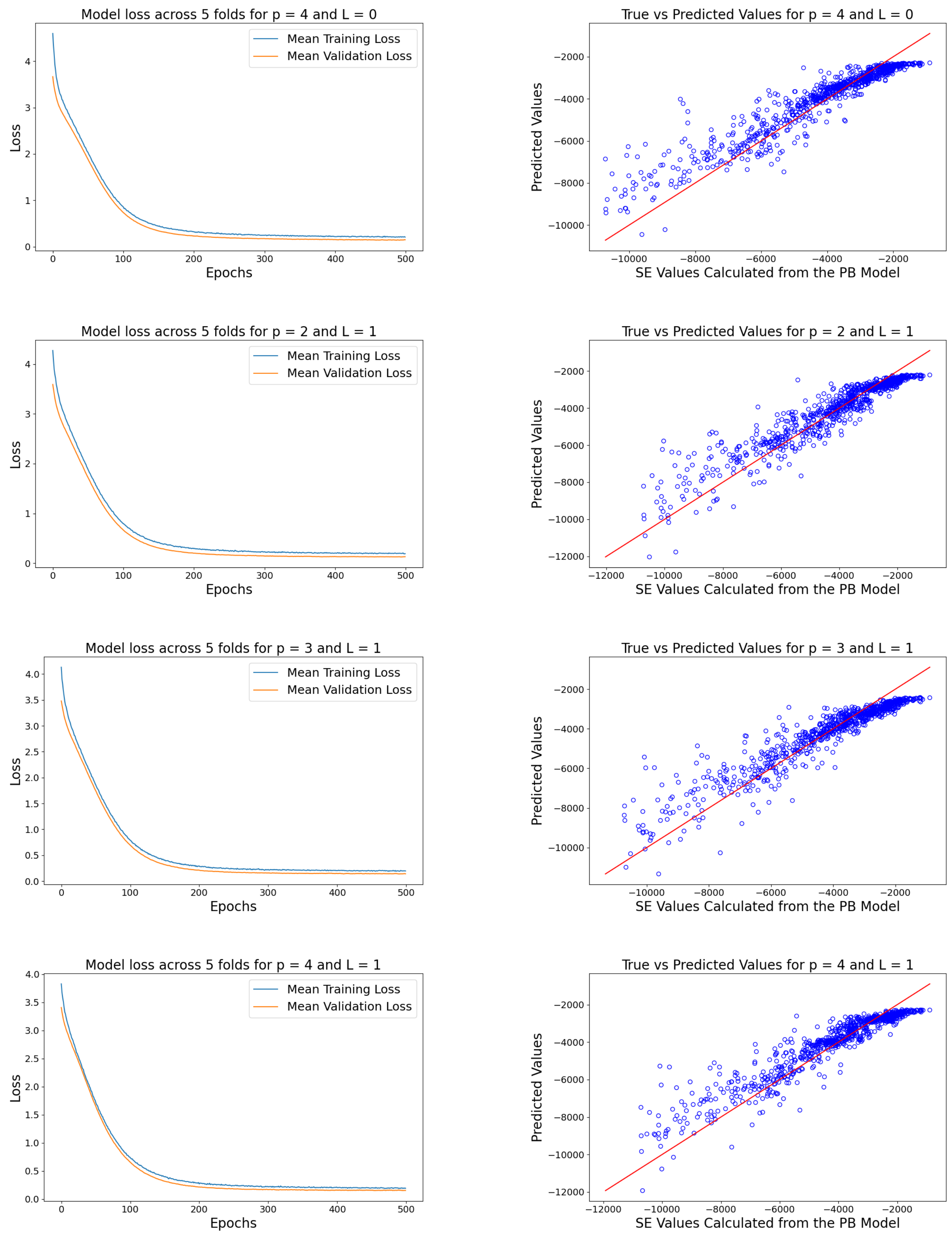


Figure 4.7: Loss plots (left) and scatter plots (right) for solvation energy computed with PB model prediction using electrostatic features for $p = 4/L = 0$, $p = 2/L = 1$, $p = 3/L = 1$ and $p = 4/L = 1$

4.2.2.3. Solvation Energy From the Generalized Born Model

In this section, we investigate the prediction task again of solvation energy but now computed by the Generalized-Born model. The results are similar to the previous section which is no surprise because, as described in Chapter 3, the Generalized-Born model approximates the Poisson Boltzmann model. The motivation is that the efficiency advantages of the GB model make it a faster alternative to the PB model to provide advanced electrostatic features such as the global solvation energy or local reaction potential at the charge sites. Figure 4.8, displays the solvation energy values calculated from both models. The Generalized Born solvation energy values are computed using a modified version of the software called GBNSR6, [**?**]. It uses MSMS to generate the molecular surface which is the same surface generator used to compute the Poisson-Boltzmann model derived solvation energy. Thus the GB and PB solvation energies are highly correlated as seen in Figure 4.8.

It is evident that the Generalized-Born model accurately approximates the Poisson Boltzmann model. Figure 4.7, displays the results in terms of the specified evaluation metrics. The best performance, in terms of the correlation coefficient, is 0.95 for $p = 4/L = 0$, $p = 2/L = 1$, $p = 3/L = 1$ and $p = 4/L = 1$ (shown in Figure 4.9). Note these are the same optimal cases for the solvation energy computed by the PB model prediction. Again, we see the loss plots in the left column of Figure 4.9 demonstrate a model with good fit without obvious signs of overfitting. The scatter plots in the right column of Figure 4.9 show similar trends as in the scatter plots in the right column of Figure 4.7. So, again we see that the curated electrostatic features play a crucial role in predicting the target label, as indicated by the loss and scatter plots in Figure 4.9.

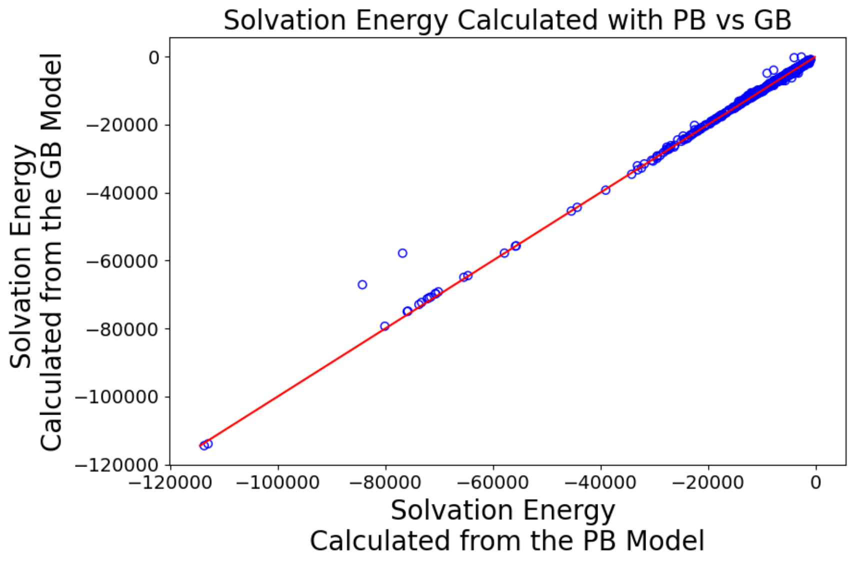


Figure 4.8: Scatter plot of Poisson Boltzmann versus Generalized Born predicted values

Table 4.7: Results for solvation energy computed by Generalized-Born prediction

p	L	$N(p,L)$	MSE	PCC	R^2	MAPE
1	0	4	0.45	0.77	0.58	0.27
2	0	10	0.22	0.91	0.80	0.20
3	0	20	0.14	0.94	0.87	0.15
4	0	35	0.13	0.95	0.88	0.15
0	1	9	0.53	0.73	0.51	0.32
1	1	36	0.20	0.91	0.82	0.18
2	1	90	0.15	0.95	0.86	0.19
3	1	180	0.16	0.95	0.86	0.15
4	1	315	0.14	0.95	0.87	0.15
0	2	73	0.30	0.85	0.72	0.23
1	2	292	0.21	0.92	0.81	0.18
2	2	730	0.24	0.92	0.78	0.18
3	2	1460	0.23	0.93	0.79	0.18
4	2	2555	0.25	0.92	0.77	0.22
0	3	585	0.40	0.82	0.63	0.23
1	3	2340	0.44	0.83	0.59	0.21

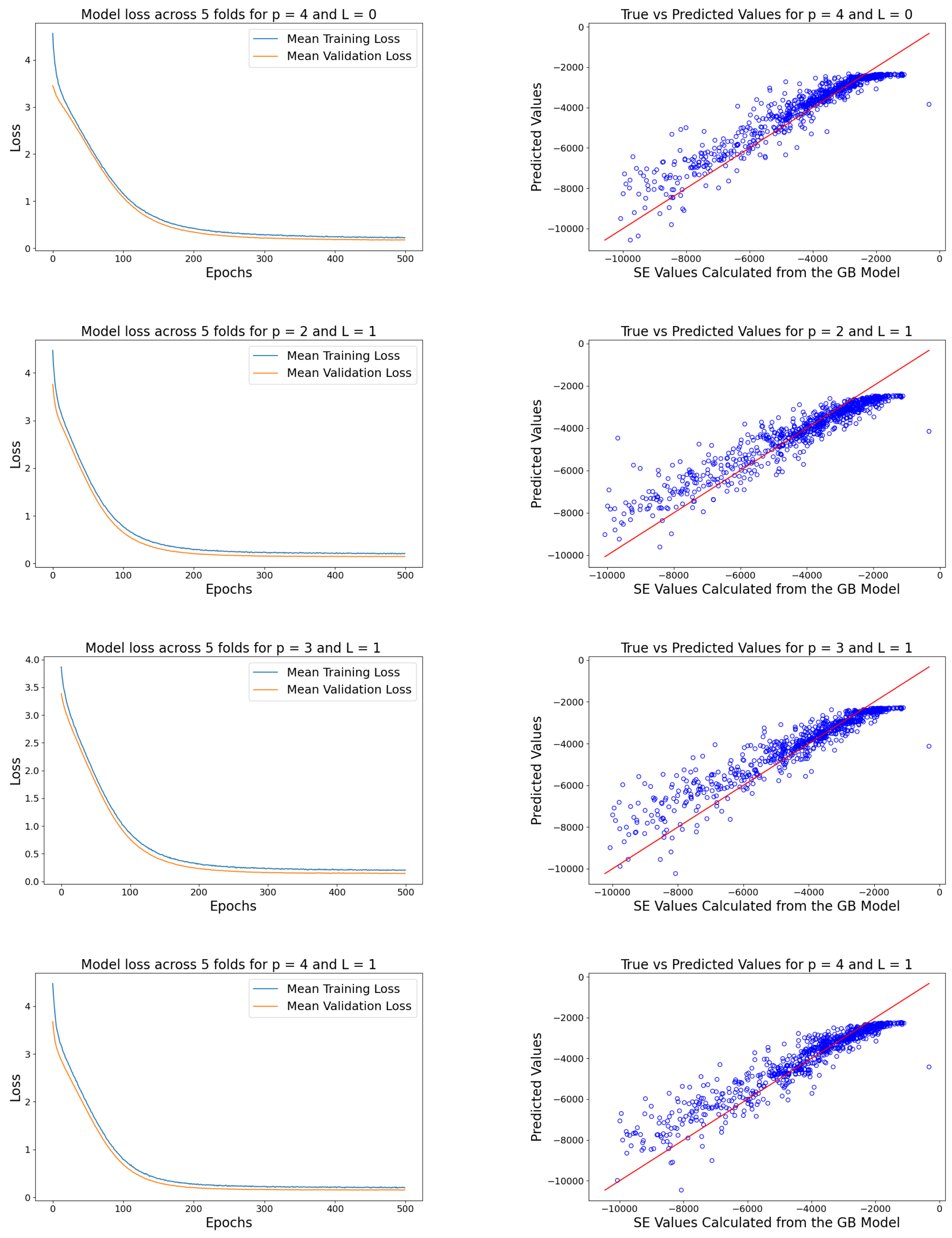


Figure 4.9: Loss plots (left) and scatter plots (right) for solvation energy computed with Generalized-Born prediction using electrostatic features for $p = 4/L = 0$, $p = 2/L = 1$, $p = 3/L = 1$ and $p = 4/L = 1$

4.2.2.4. Discussion

Here we present summary evaluation metrics and show how the model performs as $L = 1$ (left) and $L = 2$ (right) is fixed and p varies for all three prediction tasks; solvation energy predicted by the Poisson Boltzmann model labeled as ”SE w/ PB”), solvation energy predicted by Generalized Born (labeled as “SE w/ GB”) and Coulomb energy (labeled as “CE”) in Figure 4.10. Each row represents a different evaluation metric; Pearson correlation coefficient, R^2, mean squared error and mean absolute percentage error. We see that in most cases, the model performs better for the solvation energy (computed by both TABI and GB) prediction over the Coulomb energy prediction and all cases are highly dependent on the order p as L is fixed. We see a sharp increase in the correlation coefficient and R^2 as well as decrease in MSE and MAPE as p increases from 0 to 1 then slower improvements as p continues to increase to 2 for both $L = 1$ and $L = 2$.

From Figure 4.6, Figure 4.7 and Figure 4.9 the scatter plots (right column) show that the model seems to predict higher values for lower true values and predict lower values for higher true values. This may suggest a need to further tune hyperparameters such as the number of neurons, dropout rate, activation functions, and/or batch size. Each case demonstrates the notion that more accuracy in the multipole expansion (increase in p) results in improved model performance (in terms of all evaluation metrics investigated). It would be beneficial to investigate these trends further by including more p/L cases to include a higher number of features. This would require more data to limit the effects of the curse of dimensionality but would be a worthwhile experiment.

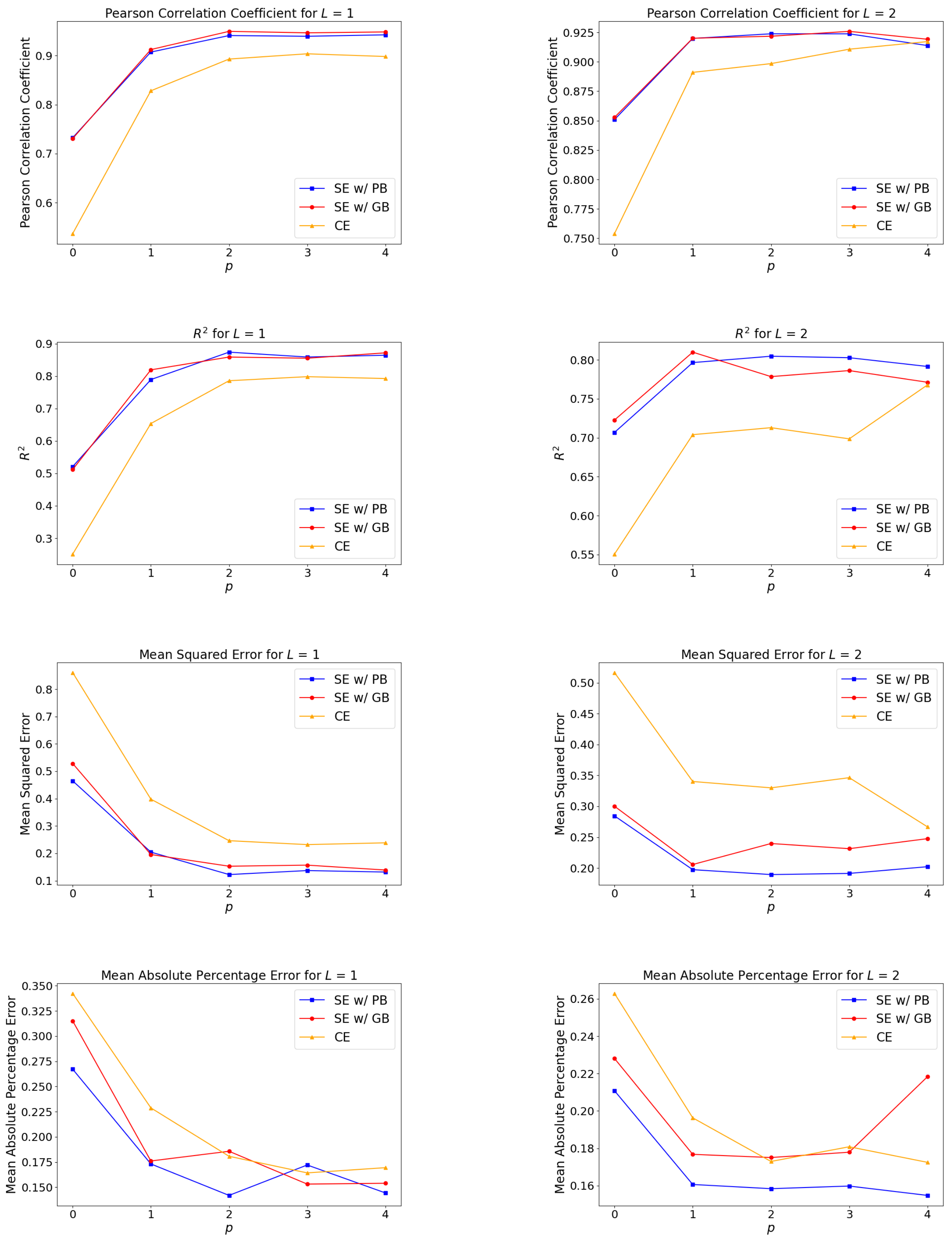


Figure 4.10: Performance of predicting Coulomb energy (CE), solvating energy using PB model (SE w/PB), solvation energy using GB model (SE w/GB) in terms of Pearson Correlation Coefficient, R^2, Mean Squared Error, and Mean Absolute Percentage Error for fixed $L = 1$ (left) and $L = 2$ (right) to highlights the effects of increasing p

4.2.3. Significance of Topological Features and Electrostatic Features

Here we revisit the use of topological features using a subset of the v2018 dataset consisting of 1178 proteins. After applying the interquartile range method to remove outliers the remaining dataset consists of 1093 proteins. Again we use 80% of the data for the training set and reserve 20% for the testing set. We use $k = 5$ fold cross-validation and the loss plots reported here are an average of the $k = 5$ folds. To avoid repetition, we focus on solvation energy computed with the PB model only.

4.2.3.1. Electrostatic Features Only

We include 9 test cases (p and L combinations) because we consider the curse of dimensionality: A large number of features relative to the number of observations (proteins) can lead to overfitting. We only include the 9 cases reported because they produce a reasonable number of features compared to the number of proteins used. In Table 4.8, we see model performance increases as L is fixed and p increases except in the case when $p = 4$ and $L = 0$ where we see an increase in MSE and decrease in PCC and R^2. This comes as a surprise because $p = 4/L = 0$ was an optimal case when predicting solvation energy in Table 4.6. We plot loss plots (left) and scatter plots (right) for the cases $p = 3/L = 0$, $p = 4/L = 0$, $p = 2/L = 1$ and $p = 3/L = 1$ in Figure 4.11. The loss plots with 500 epochs demonstrated a need to continue training, so we increased the number of epochs to 750. These plots still demonstrate a model with good fit that is able to capture the underlying patterns of the dataset.

Table 4.8: Results for Solvation Energy Computed by PB Using Solely Electrostatic Features

p	L	$N(p,L)$	MSE	PCC	R^2	MAPE
1	0	4	0.35	0.78	0.59	0.23
2	0	10	0.24	0.87	0.72	0.19
3	0	20	0.18	0.91	0.79	0.15
4	0	35	0.23	0.86	0.74	0.15
0	1	9	0.41	0.73	0.53	0.25
1	1	36	0.24	0.86	0.73	0.16
2	1	90	0.14	0.92	0.84	0.13
3	1	180	0.15	0.91	0.80	0.15
0	2	73	0.37	0.77	0.57	0.22

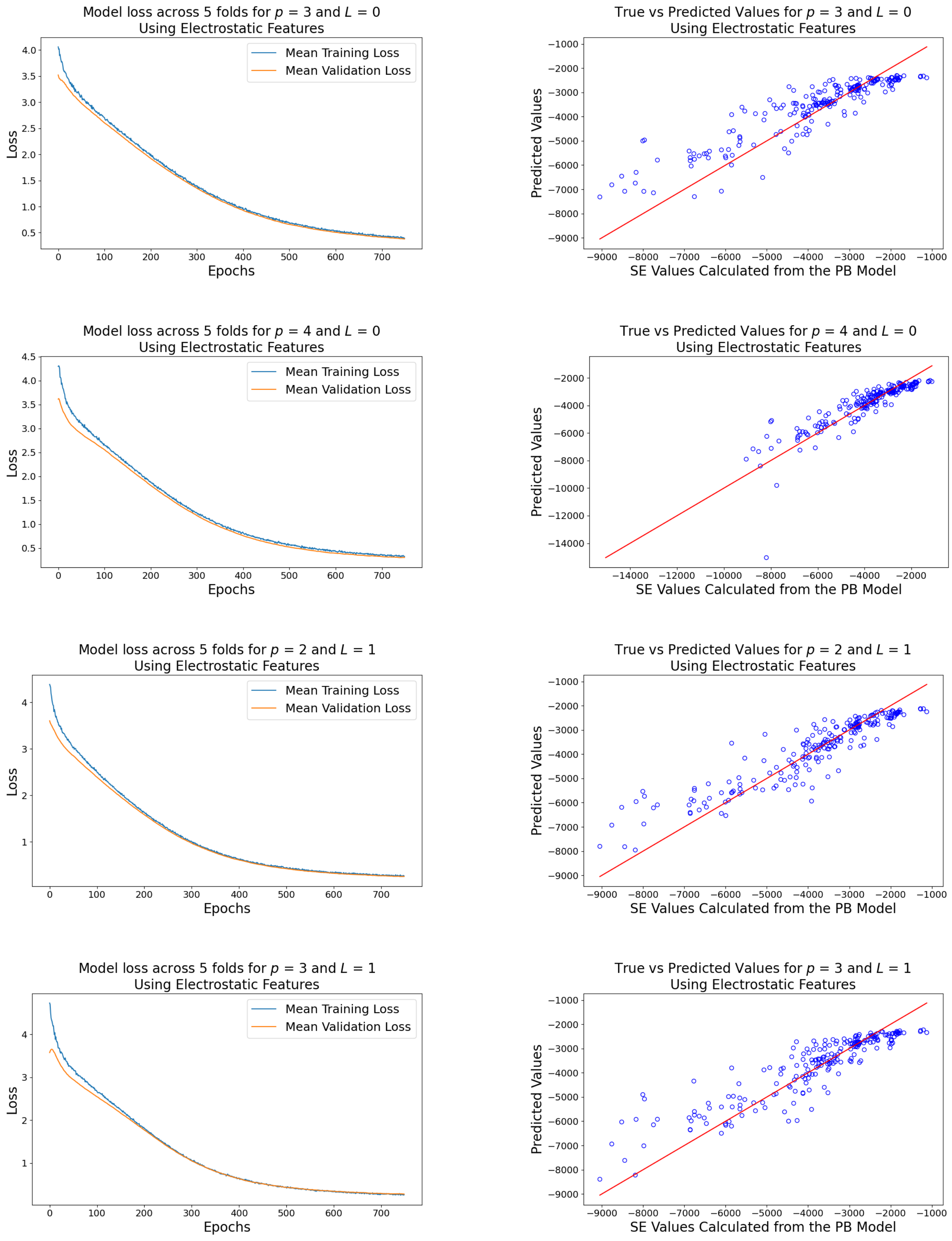


Figure 4.11: Loss plots (left) and scatter plots (right) for solvation energy computed with PB prediction using electrostatic features for $p = 3/L = 0$, $p = 4/L = 0$, $p = 2/L = 1$ and $p = 3/L = 1$

4.2.3.2. Topological Features Only

Our goal is to determine if the electrostatic features and topological features can be used together to predict solvation energy with higher accuracy than on their own. Here we test the topological features on their own. We visualize the topological features (image-like representations) in Figure 4.2.3.2 with corresponding solvation energy calculations. These images do not appear to demonstrate differences to the human eye but demonstrate distinguishable differences to the convolutional neural network (depicted in Figure 3.7). To better demonstrate the variations, we visualize the raw barcodes using the Gudhi package in Figure 4.13. We chose 2 sample proteins, 184l and 185l. According to [3], both proteins specify "ligand binding in a buried non-polar cavity of T4 lysozyme". Protein 184l has a solvation energy of -2401.28 kcal/mol and consists of 2603 atoms while Protein 185l has a solvation energy of -2434.42 kcal/mol and also consists 2603 atoms. We see that even though these proteins have similar biological functions the persistent homology approach can capture variations in their structure relevant to the prediction at hand.

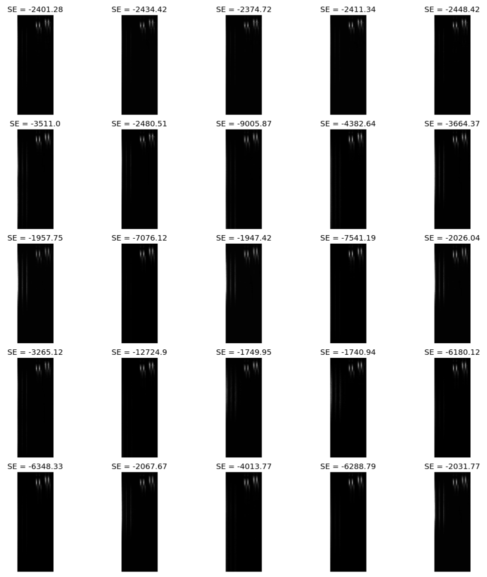


Figure 4.12: Grid of convolutional inputs with corresponding solvation energy values

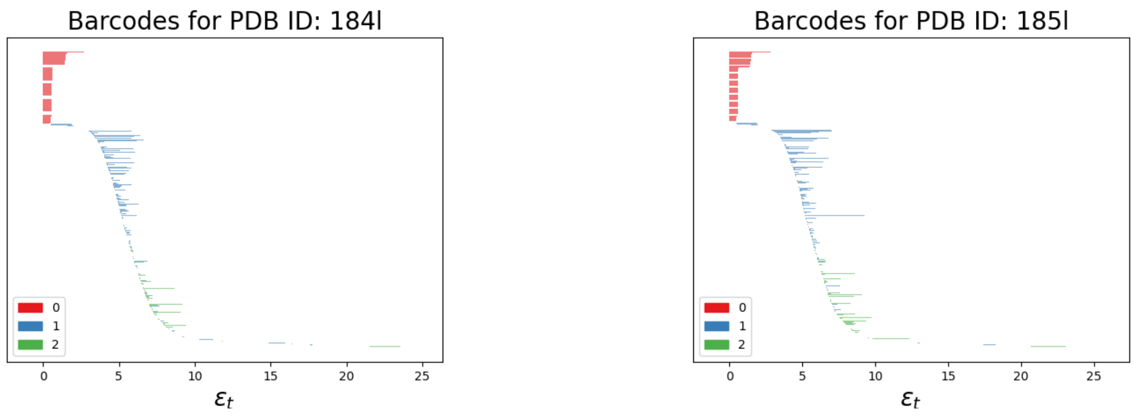


Figure 4.13: Barcodes generated in Gudhi for protein with PDB ID 184l (left) and protein with PDB ID 185l (right)

We used the model architecture described in Figure 3.7, motivated by the network used in [65] to predict binding affinity. We see in Table 4.9 and Figure 4.14, that the topological features are indeed effective at predicting solvation energy (computed by the Poisson-Boltzmann). In particular, the topological features are used to predict solvation energy with higher performance (in terms of the correlation coefficient of 0.90) compared to binding affinity for this specific dataset. We see the loss plot shows convergence for both the training and validation sets and the scatter plots indicate a high degree of correlation between the model's predicted values and the true values, but we do see variability around the line $y = x$, suggesting the need for improvement.

Table 4.9: Evaluation metrics for the model using solely topological features to predict solvation energy computed by the PB model

MSE	PCC	R^2	MAPE
0.16	0.90	0.82	0.12

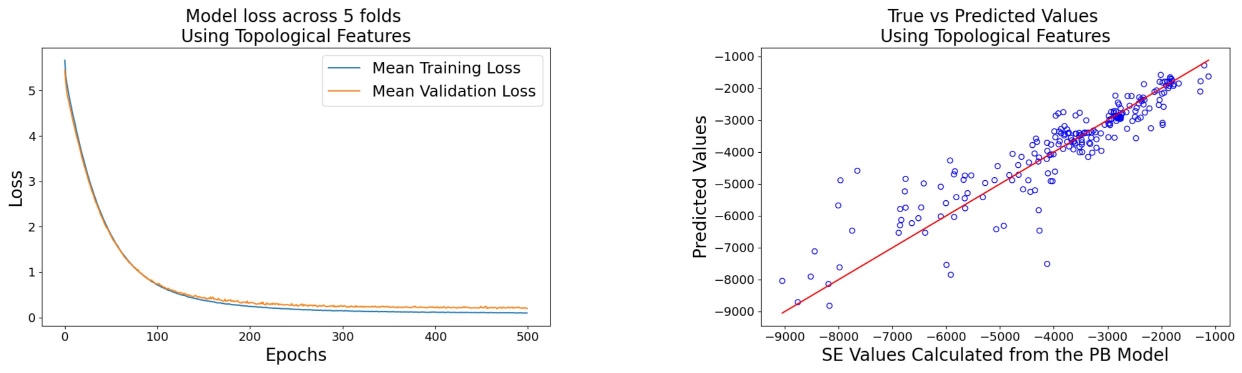


Figure 4.14: Loss plot (left) and scatter plot (right) for solvation energy computed with the PB model prediction using topological features

4.2.3.3. Both Electrostatic and Topological Features

Now that we have established results to predict solvation energy separately using electrostatic features and topological features we test the merged model presented in Figure 3.8. The merged model using both datatypes demonstrates improved performance compared to the topological model (Figure 3.7) and electrostatic model (Figure 3.6) alone. In Figure 4.10, each case of p and L shows higher Pearson correlation coefficients compared to each case in Table 4.8. Each case in Figure 4.10 also outperforms the model using solely topological features. This suggests that the topological features are benefited from the incorporation of electrostatic features and conversely, the electrostatic features benefit from the addition of the topological features. We plot the best cases in Figure 4.15. The loss plots demonstrate convergence and the scatter plots reduce the variability from the line $y = x$ compared to the plots in Figure 4.11.

Table 4.10: Results for solvation energy computed by PB using *both electrostatic and topological features*

p	L	$N(p,L)$	MSE	PCC	R^2	MAPE
1	0	4	0.09	0.96	0.89	0.10
2	0	10	0.06	0.96	0.92	0.09
3	0	20	0.13	0.93	0.84	0.11
4	0	35	0.14	0.93	0.84	0.09
0	1	9	0.11	0.94	0.87	0.11
1	1	36	0.09	0.95	0.90	0.10
2	1	90	0.11	0.94	0.87	0.10
3	1	180	0.15	0.92	0.83	0.11
0	2	73	0.13	0.92	0.84	0.11

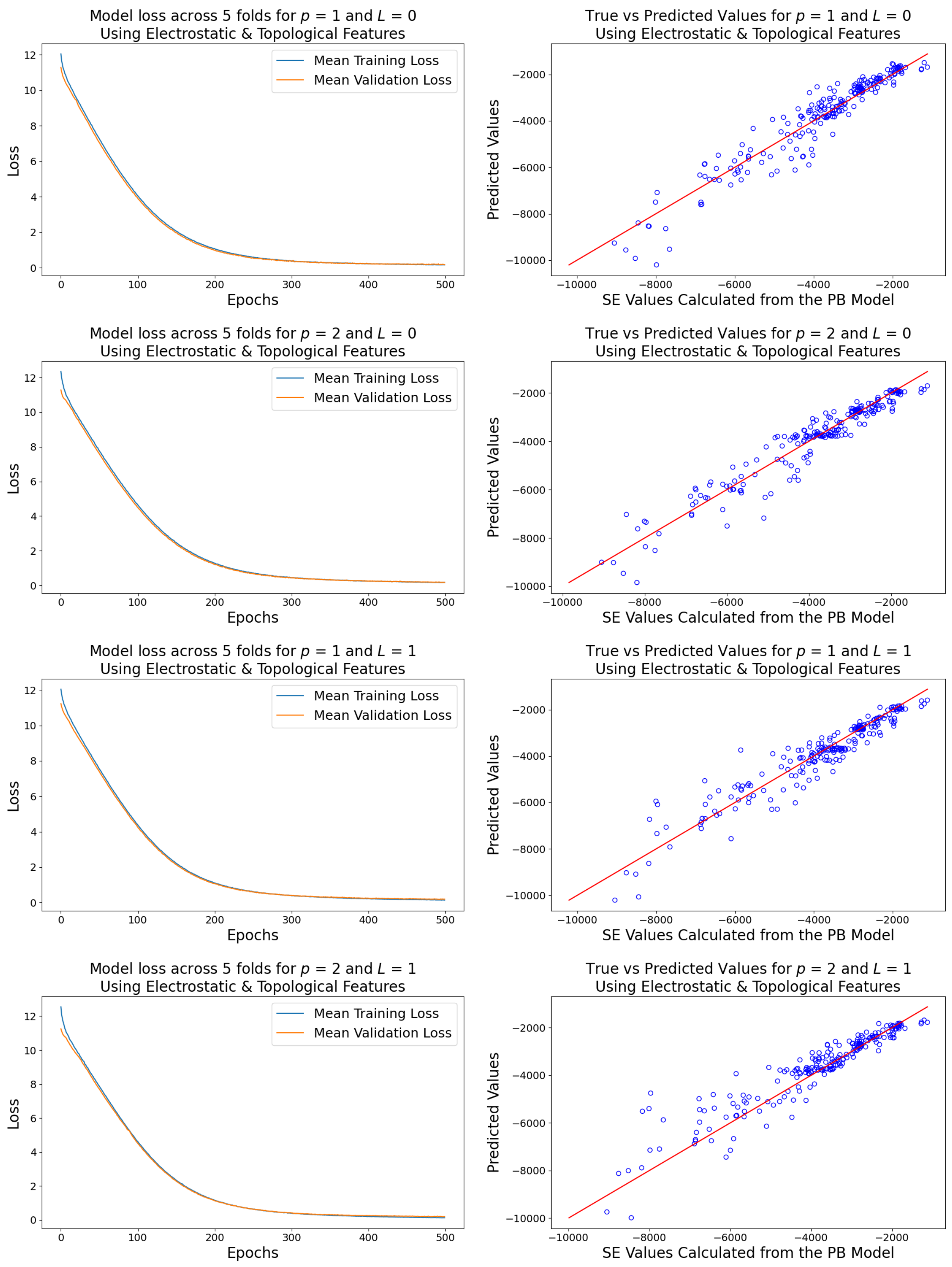


Figure 4.15: Performance using both electrostatic and topological features for selected electrostatic parameters p and L.

4.2.3.4. Discussion

Figure 4.16, shows the evaluation metrics for fixed $L = 0$ (left) and $L = 1$ (right). In each graph, the red line represents the results using both electrostatic and topological features (architecture in Figure 3.8), the blue line shows the result using electrostatic features (architecture in Figure 3.6), the yellow line shows the result using topological features (architecture in Figure 3.7). Note that the model which uses solely topological features does not depend on p or L, but is included in these plots as a reference to demonstrate that the combined model outperforms both models using topological and electrostatic features alone for all evaluation metrics considered. Given that both methods extract information from the complex 3-dimensional structure of proteins in very different ways, but merged together show improved model performance suggests a benefit of our approach. This motivates the use of various types of features, from different areas of mathematics, in combination for better performance of biological quantities relevant to drug discovery/design.

From Figure 4.15 the scatter plots (right column) show that the model still has some variability and has room for improvement which suggests a need to further tune hyperparameters such as the number of neurons, dropout rate, activation functions, and/or batch size. It would be a worthwhile experiment to use grid search to identify the optimal parameters to achieve even better performance.

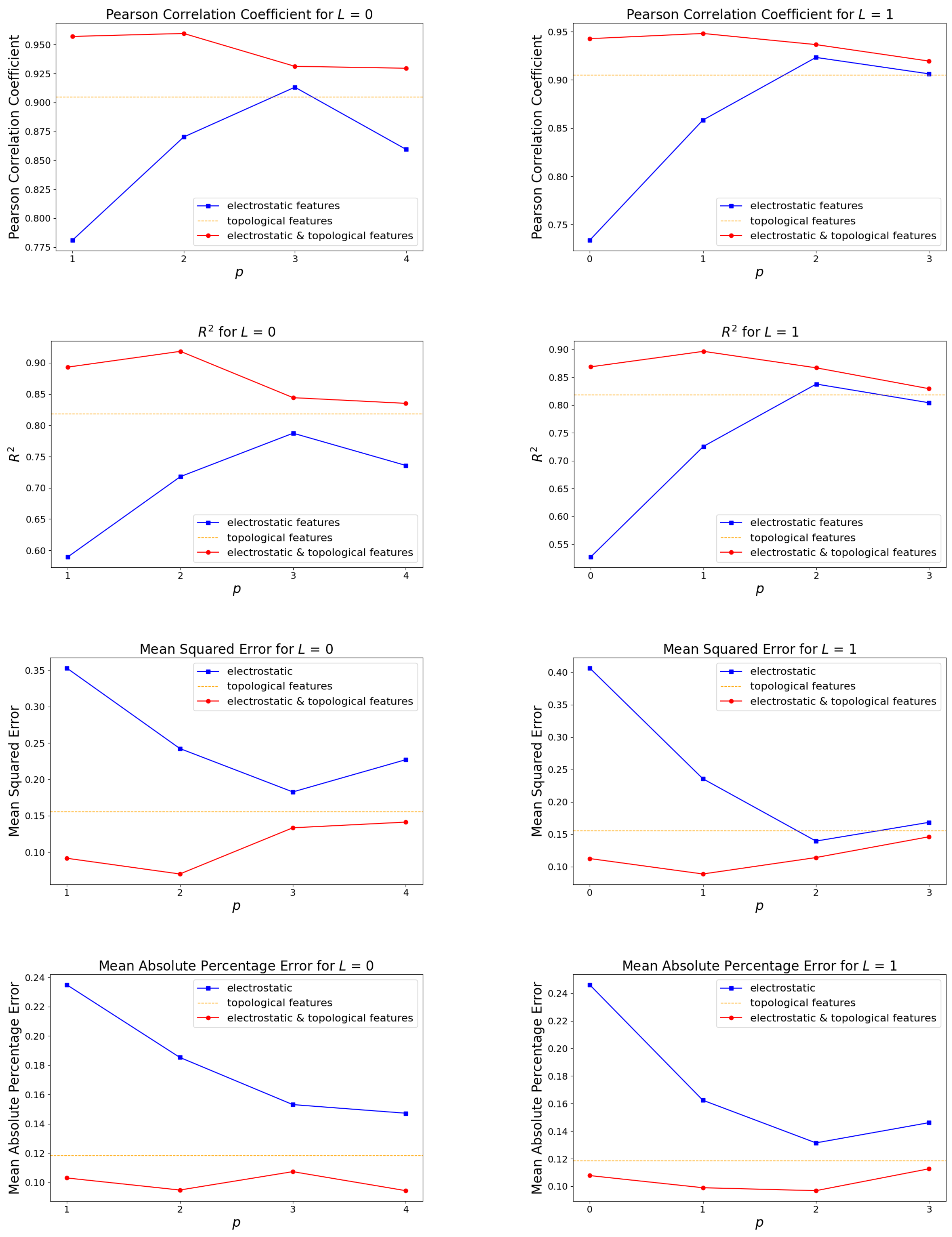


Figure 4.16: Comparison plots in terms of Pearson Correlation Coefficient, R^2, Mean Squared Error, and Mean Absolute Percentage Error for fixed $L = 1$ (left) and $L = 2$ (right) to highlight the effects of increasing p

Chapter 5
Cancer Research

5.1. Trial Design and Methodology

Tumor annotated DNA polymorphism data was generated by Ocean Ridge Biosciences (ORB) (Deerfield Beach, Florida), across 981 validated genes for all patients who entered into the Phase IIb double-blind randomized placebo-controlled trial (NCT02346747) comparing Vigil and placebo in Stage III/IV resectable ovarian cancer. Patient demographics, trial design, and vaccine manufacturing were previously described in [26]. Patients were enrolled following IRB-approved written consent. DNA samples of malignant tissue were analyzed from all 91 patients entered into trial and results were compared to clinical endpoints prospectively identified in the study statistical plan. Gene variants were classified by either Ingenuity Variant Analysis software (Qiagen, Valencia, CA) or the NIH ClinVar database (current versions as of 10 February 2020) [75]. Only gene variants that were determined to be pathogenic or likely pathogenic were included and were referred to as pathogenic mutations. Individual gene sets of pathogenically mutated genes for each patient in the trial were generated. An overall gene set was then constructed by taking the union of the individual gene sets. A binary mutation matrix was constructed from this overall gene set such that element (i, j) of the mutation matrix was equal to 1 if patient i had a pathogenic mutation in gene j and equal to 0 if patient i was wild type in gene j.

5.1.1. STRING and Topological Distance

The STRING (Search Tool for the Retrieval of Interacting Genes/Proteins) database has been maintained since 2000 by the Swiss Institute of Bioinformatics, CPR - Novo Nordisk Foundation Center Protein Research, and EMBL - European Molecular Biology Laboratory [76] Pathogenically mutated genes were inputted into the STRING application (Version 1.5.1) in Cytoscape (Version 3.8.0) to gauge functional interaction. The STRING application generates a network for the input genes which consists of genes and their interactions, represented by nodes and the lines which connect them, referred to as vertices and edges, respectively. Genes are only connected via edges in the network if there is evidence they interact from published literature and high throughput experimental data. STRING uses this information to assign confidence scores, which are denoted as $s(i,j)$, to each interaction or edge. Individual STRING scores are produced for each of the interaction types and these scores are integrated to give a combined confidence score, $s(i,j)$, between each pair of proteins. Each protein-protein interaction (PPI) score is bound between 0 and 1 which indicates how likely STRING judges the particular interaction to be true, given available evidence. Next, edge weights (w) between each pair of genes are calculated according to the following formula: $w(i,j) = 10(1 - s(i,j))$. The score, $s(i,j)$ was subtracted from one so that intuitively a small weight corresponds to strong evidence of a biological interaction between a gene pair and multiplied by 10 to shift the values to the desired scale [77]. Dijkstra's Algorithm was then used to calculate the length of the shortest weighted path between genes denoted, $d(i,j)$ by summing over the weighted edges that connect them and systematically finding the shortest weighted path. Genes with distance ≤ 3.8 was defined as the bottom quarter. Intuitively, when a gene pair has a low topological distance, $d(i,j)$, the genes may interact biologically.

5.1.2. C-scores

The independent concepts of patient mutation profiles and the STRING Network are integrated by C-score. The probability of co-mutation for every pair of genes in the overall gene set was calculated. The probability (P) of a co-mutation was defined as the total number of the 91 patients who have a mutation in both of the genes in the given gene pair divided by a measure of the total number of times both genes are mutated individually, given by:

$$P(i,j) = \frac{|G(i) \cap G(j)|}{\sqrt{m(i) * m(j)}}$$

[78] . Where $|G(i) \cap G(j)|$ represents the number of individual tumors where both genes i and j are mutated, and $m(i)$ and $m(j)$ are the cumulative mutations of genes i and j, respectively. The range of $P(i,j)$ is between 0 and 1 where $P(i,j) = 0$ indicates that genes i and j never co-mutate and $P(i,j) = 1$ means the genes always co-mutate. The probability of co-mutation and the topological distance between genes in the STRING network were then combined to calculate a C-score, denoted $C(i,j)$, to quantify the likelihood that the genes interact functionally, termed "putative genetic interactions" [78]. The C-score is calculated by dividing the probability of co-mutation by the topological distance from the STRING network squared.

$$C(i,j) = \frac{P(i,j)}{d(i,j)^2} = \frac{|G(i) \cap G(j)|}{\sqrt{m(i) * m(j)}d(i,j)^2}$$

Further, the cumulative C-score for a gene i is denoted, cum$C(i) = \sum_{i \neq j} C(i,j)$ [78]. A gene with a high cumulative C-score is more likely to co-mutate with genes close to it in the STRING Network. To determine the significance of these C-scores, a permutation test is performed. We began by reshuffling the mutation profile of each patient by preserving the number of mutations of each patient and randomly assigning new mutations. Then we followed the above methodology to calculate "simulated C-scores", $C_s(i,j)$. The p-value was then calculated by taking the total number of times the simulated C-score was greater than or equal to the actual C-score and dividing by the number of trials performed ($n = 10,000$) [78].

$$p = \frac{|C_s(i,j) \geq C(i,j)|}{10,000}$$

.

5.1.3. Pathway Analysis

A list of pathways associated with each gene was extracted and binned into seven color-coded categories that included DNA repair, chromosomal organization and transcription, regulation of translational and post-translational modification, immunity, other pathways, other cancer genes, and undefined.

5.1.4. Survival Analysis

RFS and OS relationships of patients with varying mutational statuses in

1. hub genes
2. gene pairs with small topological distances
3. gene pairs with high cumulative C-scores

were explored. Finally, patients were stratified by mutation statuses mutant versus wild type to examine and compare RFS and OS differences using the 'survival' and 'survminer' packages in R (Version 3.6.2).

5.2. Results

5.2.1. Network Construction and Pathway Enrichment Analysis

To identify potential gene interactions that were associated with extended RFS in patients receiving Vigil, first a protein-protein interaction (PPI) network was constructed using 83 genes that were identified as having pathogenic mutations in the study population [79]. The 83 genes were loaded into STRING software; 77 of these genes were identified as having functional data in the database and were used to construct a network. The six genes that STRING did not recognize include *AC092143.1*, *AL132855.1*, *MHRT*, *MYCN*, *NBR2* and *ZFPM2-AS1*. These were not included in the network or the degree chart. The STRING-constructed PPI network is displayed in 5.1. In the STRING network, an association or interaction may refer to direct (e.g., physical binding) or indirect interactions, such as shared participation in a common metabolic pathway [80]. Nodes in the network represent genes ($n = 77$) and edges ($n = 371$) represent biological interaction or association between any two of the identified pathogenic genes. Pathway analysis was conducted by inputting each gene in the STRING network into the WikiPathways Application in Cytoscape [79].

Figure 5.1 is the STRING Network produced in Cytoscape. Larger nodes indicate hub genes. A node colored red indicates a majority of pathways involved with DNA repair. Blue indicates chromosomal organization and transcription. Purple indicates regulation of translation and post-translational modification. Yellow indicates immunity. Green indicates other cancer genes. Gray genes are those that had no known pathways in WikiPathways orange genes do not fall into the other six categories.

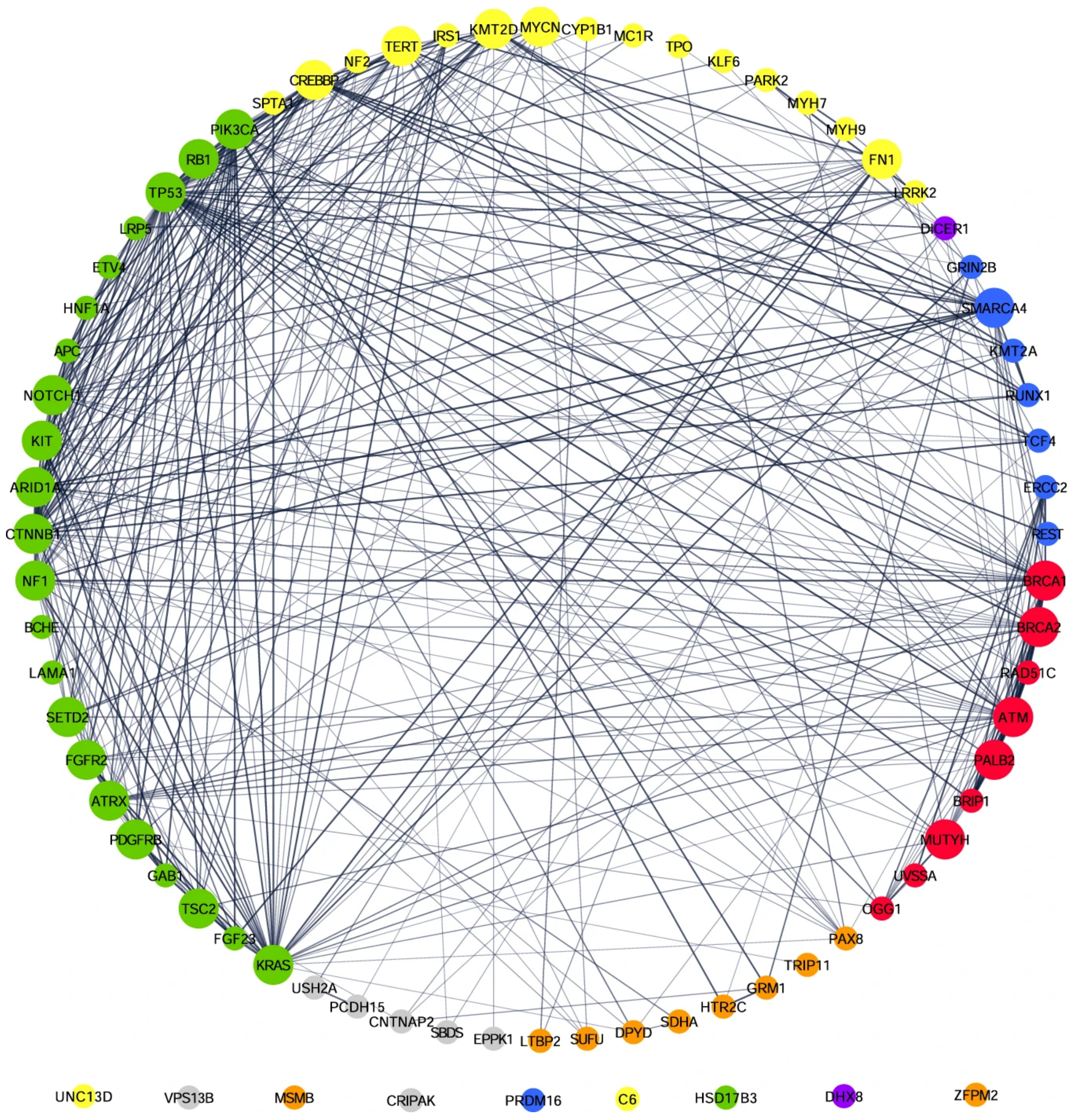


Figure 5.1: String Nework - Larger nodes indicate hub genes. A node colored red indicates a majority of pathways involved with DNA repair. Blue indicates chromosomal organization and transcription. Purple indicates regulation of translation and post-translational modification. Yellow indicates immunity. Green indicates other cancer genes. Gray genes are those which had no known pathways in WikiPathways orange genes do not fall into the other six categories.

5.2.2. Single Gene Analysis

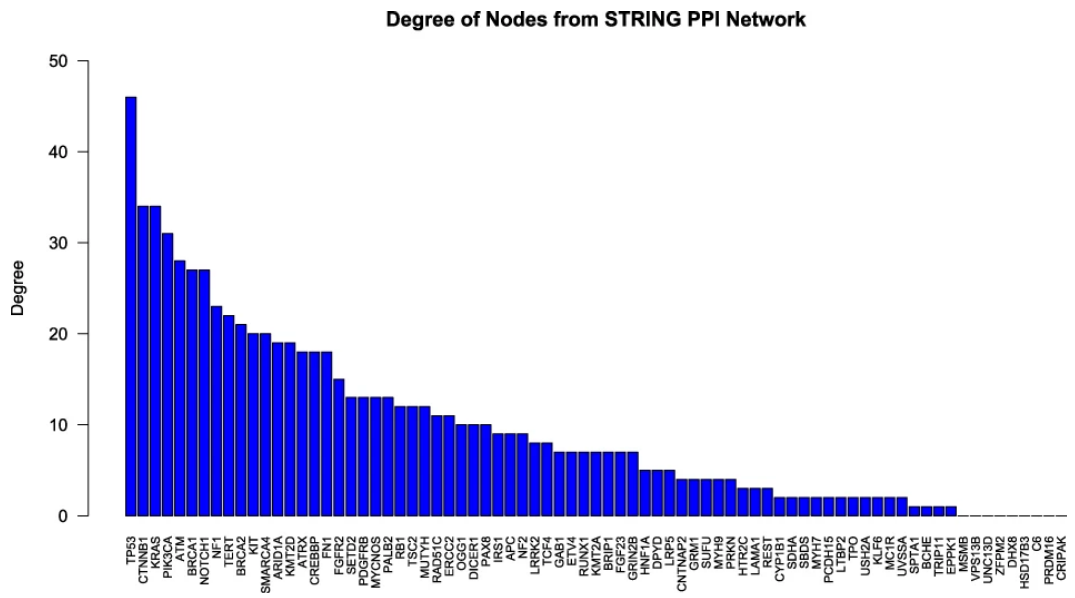


Figure 5.2: Degree of nodes from STRING PPI Network. Each gene in the STRING PPI Network is represented from largest to smallest degree below.

A hub gene is defined to be a gene with a high degree (or number of connections) of associations with other genes in the network. Here, a hub gene is considered to be a gene with degree ≥ 12, which is the top quartile of genes based on the range of degree of genes in the network. Ten of the 23 genes in the distance matrix (Figure 5.3) were identified to have a degree ≥ 12, in the network defining them as hub genes: *TP53*, *CTNNB1*, *PIK3CA*, *BRCA1*, *NF1*, *BRCA2*, *ARID1A*, *ATRX*, *MYCNOS*, and *MUTYH*. *TP53* had the largest degree of all genes in the network as seen in Figure 5.2. For the single gene analysis, patients were grouped by their mutational status and the Kaplan-Meier and log-rank tests were performed to determine whether the mutational status at that loci was associated with RFS in Vigil-treated patients relative to placebo. RFS from randomization was the primary endpoint

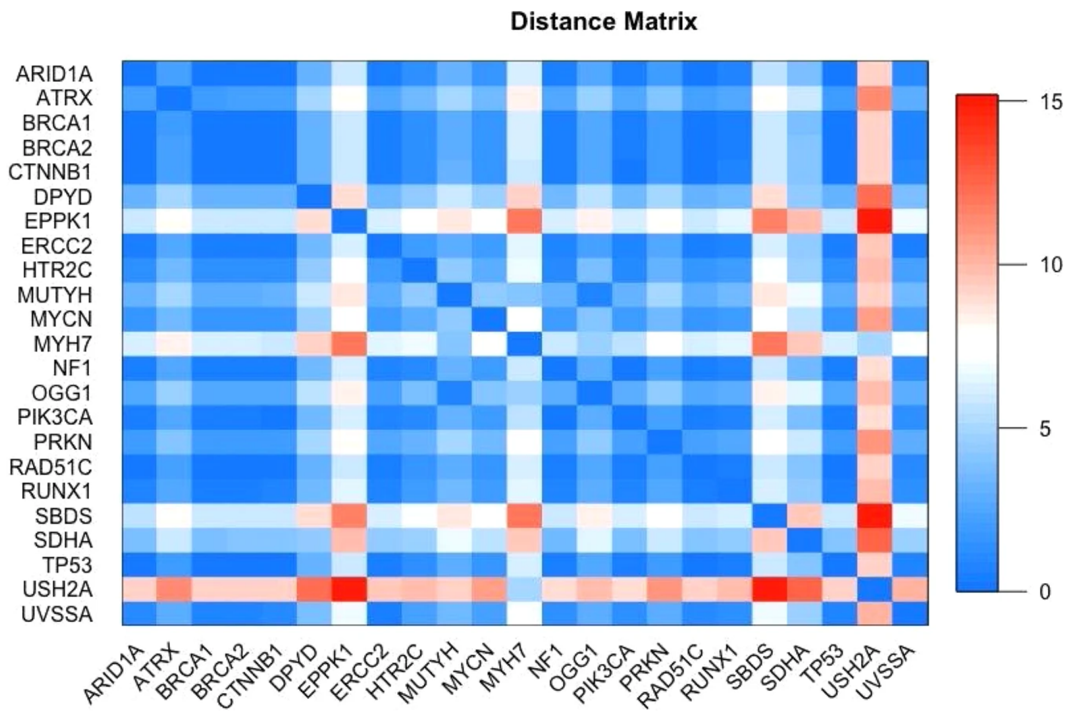


Figure 5.3: Heat map key is located to the right of the matrix. Blue shading represents a high degree of interaction while red represents less interaction. The diagonal dark blue shade is a genes interaction with itself and is equal to 0.

of the VITAL trial and therefore it was used as the test variable (or categorical variable) for this analysis. *TP53*, *BRCA1*, and *BRCA2* were the only hub genes that reached our planned statistical cutoff of $p \leq 0.1$ from randomization (Table 5.1). In patients with *TP53*-mutated tumors (*TP53*m; $n = 65$), median RFS was 18.69 months with Vigil ($n = 33$) and 8.35 months with placebo ($n = 32$) (one-sided $p = 0.096$, HR = 0.66). In the *BRCA1*wt population ($n = 79$), RFS was 12.75 months for Vigil ($n = 42$) and 8.38 months for placebo ($n = 37$) (one-sided $p = 0.10$, HR = 0.70). RFS for the *BRCA2*wt population ($n = 82$) was 11.47 months and 8.35 months for Vigil ($n = 46$) and placebo ($n = 36$), respectively (one-sided $p = 0.05$, HR = 0.64).

5.2.3. Gene Pair Analysis

5.2.3.1. Gene Pairs with a Small Topological Distance

The distance matrix in Figure 5.3 only included 23 genes, representing a subset of the network that was complete. Once the complete network was found, the distance matrix was constructed to visualize the topological distance of each corresponding gene pair. The maximum and minimum distance in the matrix were between genes *USH2A* and *EPPK1*, with a distance of 15.19 and between genes *BRCA1* and *BRCA2* with a distance of 0.02, respectively. We further analyzed gene pairs with a small topological distance. Using a cutoff of 3.8 which was calculated by taking the range, 15.19 minus 0.02, and dividing by four, denoted the bottom quartile, we arrived at 139 gene pairs comprised of 23 genes (Figure 5.3). We found that *TP53* and *BRCA1* had a distance of 0.04 and *TP53* and *BRCA2* had a distance of 0.06. This indicated that there was strong evidence that *TP53* had a functional association with *BRCA1* and *BRCA2* and thus considered both *BRCA1* and *BRCA2* as a joint relationship designated as *BRCA* which is consistent with prior analysis by others [81].

5.2.3.2. Gene Pairs with High Cumulative C-Scores

After computing the C-Score for all gene pairs in the Distance Matrix and ordering genes from highest to lowest cumulative C-Score (cumC-score), the genes with highest cumC-score in order were *BRCA1*, *BRCA2* and *TP53*. This indicated high connectivity within the network both from a co-mutation standpoint and a topological distance perspective. C-score significance analysis was performed, however due to the limitations of small sample size and small gene sets the results were not significant. Future analysis with larger sample size and gene panels is warranted. The function or dysfunction of *BRCA1*, *BRCA2*, and *TP53* has broad-reaching consequences for the other proteins within the ovarian cancer cell, more so than other genomic variants. This warranted closer attention to the effects of wild-type versus mutant expression. Given the proximity of *TP53* with *BRCA1* and *BRCA2* in the STRING network and their high cumulative C-scores, we performed survival analysis across the four mutation statuses (co-mutant, mutant-wild-type, wild-type mutant and co-wild type). The impact of these combinations on relapse-free survival in the Vigil treatment group compared to placebo is displayed in Table 5.2. The *TP53*m-*BRCA*wt group experienced a median RFS of 19.35 months, compared to 11.71 months in co-mutant and 10.48 months in co-wild type. When compared between treatment arms, the *TP53*m-*BRCA*wt group had a median RFS of 19.35 months in the Vigil arm compared to 7.85 months in placebo ($p = 0.01$, HR = 0.44; Figure 5.4). Additional tests demonstrated statistical significance ($p < 0.05$) for the combination of BRCA and our identified hub genes, displayed in Table 5.3.

5.2.3.3. Homologous Recombination Status

KM analysis was conducted to determine the effect of homologous recombination status on the *TP53*m-*BRCA*wt population. A score of < 42, as defined by Myriad Genetics was used to identify patients who were HRP and a score of ≥ 42 indicated patients were HRD. RFS in the *TP53*m-*BRCA*wt and HRP group was improved to 21.1 vs. 5.6 months (HR =

0.26, $p = 0.001$) in Vigil vs. placebo patients (Figure 5.5a) OS was also improved in HRP and *TP53*m-*BRCA*wt patients from randomization. In the Vigil treated group, OS was not reached while placebo was 27.0 months (HR = 0.33, $p = 0.02$; Figure 5.5b).

Gene	**Vigil RFS (m)**	**Placebo RFS (m)**	**Difference (m)**	**N Vigil**	**N Placebo**	**p-value**	**HR**
TP53m	18.69	8.35	10.35	33	32	0.096	0.66
BRCA1wt	12.75	8.38	4.37	42	37	0.10	0.70
BRCA2wt	11.47	8.35	3.12	46	36	0.05	0.64

Table 5.1: RFS from randomization for *TP53*m, *BRCA1*wt, and *BRCA2*wt. RFS (m) denotes median RFS in months.

Gene 1	**Gene 2**	**Vigil RFS (m)**	**Placebo RFS (m)**	**Difference (m)**	**N Vigil**	**N Placebo**	**p-value**	**HR**
BRCAm	*TP53m*	11.71	14.75	-3.04	4	14	0.27	1.50
BRCAwt	*TP53m*	19.35	7.85	11.50	29	18	0.013	0.44
BRCAm	*TP53wt*	10.48	31.90	-21.42	3	3	0.39	1.40
BRCAwt	*TP53wt*	10.48	8.38	2.10	11	9	0.47	1.05

Table 5.2: RFS from randomization for *TP53* and *BRCA*. RFS (m) denotes median RFS in months.

5.3. Discussion

Our network-based analysis of pathogenic gene mutations points us toward a potential optimally responsive population to Vigil, specifically, patients with a HRP malignant cell profile including *BRCA*wt and *TP53* mutant gene signals. These results are only hypothesis generating, but suggest a novel methodological/computational approach to biomarker assessment and for optimizing a target population for Vigil therapy and possibly proof of principle for biomarker assessment of other target-based therapies. Further evaluation of other hub genes (*PIK3CA*wt, *NF1*wt, *ARID1*wt, *MYCNOS*wt, and *MUTYH*wt) in *BRCA1/2*wt, HRP cancer patients as potential biomarkers for Vigil treatment, and possibly indicators of novel added therapeutic management, may be fruitful. In our approach, the STRING database

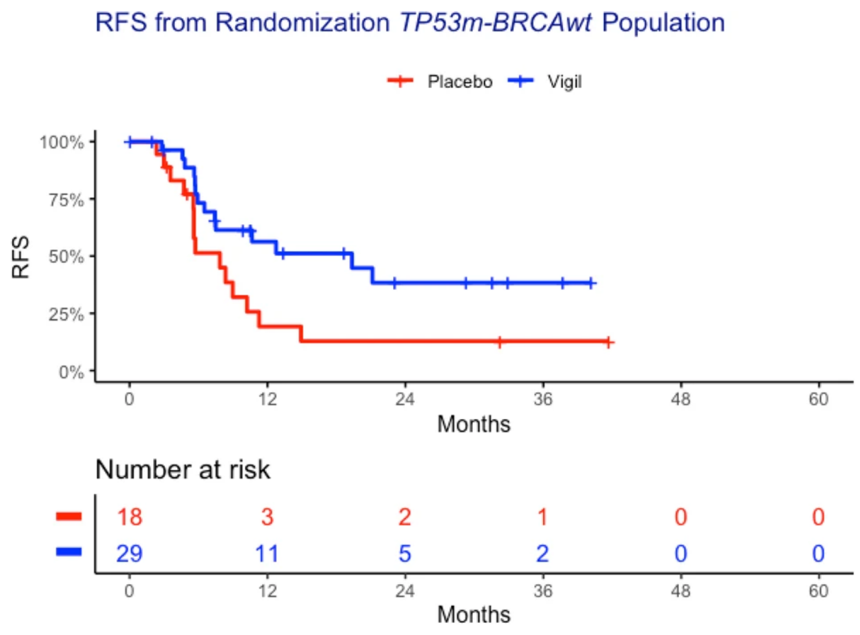
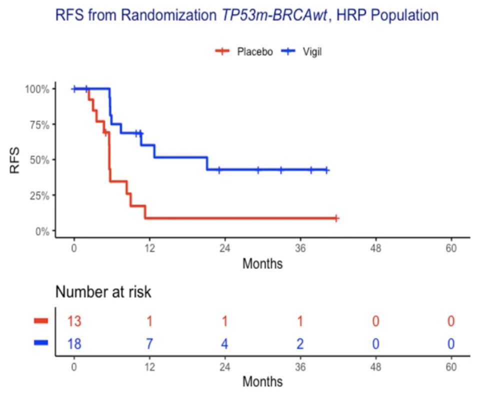


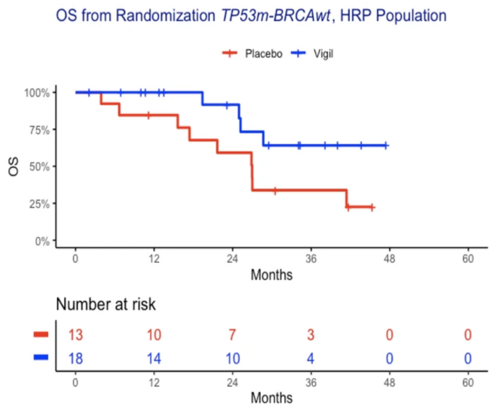
Figure 5.4: Kaplan–Meier curves of *TP53*m-*BRCA*wt population RFS from time of randomization. Vigil demonstrates RFS advantage (HR = 0.44, $p = 0.01$) in the *TP53*m-*BRCA*wt population.

Gene 1	**Gene 2**	**Vigil RFS (m)**	**Placebo RFS (m)**	**Difference (m)**	**N Vigil**	**N Placebo**	***p*-value**	**HR**
BRCAwt	*PIK3CAwt*	11.71	7.95	3.52	39	26	0.02	0.53
BRCAwt	*NF1wt*	12.75	8.35	4.40	40	26	0.04	0.59
BRCAwt	*ARID1Awt*	12.75	7.95	4.80	37	24	0.05	0.59
BRCAwt	*MYCNOSwt*	13.67	5.72	7.95	27	17	0.03	0.47
BRCAwt	*MUTYHwt*	12.75	7.95	4.80	38	27	0.03	0.56

Table 5.3: RFS from randomization for *BRCA*wt and other hub genes of significance. RFS (m) denotes median RFS in months.



(a) Vigil demonstrated RFS (HR = 0.26, p = 0.001)



(b) Vigil demonstrated OS (HR=0.33, p=0.02)

Figure 5.5: KM curves of *TP53*m-*BRCA*wt, HRP population from randomization

was utilized to construct an unbiased network to describe the functional similarity between genes, thereby providing a mechanistic understanding of the potential effect of wild type or mutant variants. This approach circumvents a potential limitation of DNA variant data and may provide more effective target population identification, given our current limited understanding of comprehensive molecular signal expression pathways and relationship to clinical benefit impact. In this manner, one can describe the genes of high importance by computationally analyzing properties of the malignant network, such as the topological distance between genes, C-scores, and hub genes. Through these analyses in the HRP ovarian population treated with Vigil, three gene variants stood out across all analytic methods: *BRCA1*, *BRCA2*, and *TP53*. The combination of STRING-generated topological distance and sample-derived probability of co-mutation is manifested as the C-score for two genes, and the cumC-score is the aggregate of one gene's interaction with every other gene in the network. Gene pairs identified by C-scores often involve central cancer genes which correlate with increased tumorigenesis and sensitivity/ resistance to anticancer therapeutics [78]. Due to the limited size of our gene set and patient sample size, we chose to characterize individual genes by their cumC-score. Here, we identified *TP53*, *BRCA1*, and *BRCA2* as genes with the highest cumC-scores of the genes present in patient samples. The high cumC-score suggests that particular variants of these genes correspond to drug response, as cumC-scores correlate with sensitivity and resistance [78]. Indeed, we found that *TP53*m and *BRCA*wt correlated with increased RFS benefit to Vigil, although further prospective analysis, now underway, will be required for verification.

Analysis of hub genes similarly identified *BRCA* and *TP53* as central ovarian cancer genes. Previous work demonstrated that hub gene data provided clinical insight to differences in OS. In one previous study, 4 of 16 identified hub genes in the studied sample (*CCNB1*, *CENPF*, *KIF11*, and *ZWINT*) were associated with decreased OS of patients with ovarian cancer [82]. Authors of this study posit that mutations in these hub genes, which occupy the intersection of many cellular pathways, results in rippling dysregulation of numerous cellular functions. Thus, by altering a single hub gene, cellular homeostasis may be impacted on a larger scale. This disruption may be associated with tumor progression, immune inhibition, and any number of cancer hallmarks, which may explain the association of hub genes with a poor prognosis [83, 84]. The hub gene analysis presented in our paper identified *TP53*m, *BRCA1*wt, *BRCA2*wt as core hub genes with RFS advantage in Vigil treated patients, potentially indicating a broader genetic network for target population of Vigil. Moreover, this approach supports a strategic shift in targeted therapeutic development towards targeting related network genomic variants.

Our results also support that the pathways impacted by *BRCA* must be intact for Vigil to function optimally, while the pathways impacted by *TP53* may be dysregulated. Similarly, the integrity of the homologous repair pathway and its associated genes (HRP genotype) may also be important for optimal Vigil results. This suggests a cancer homeostasis formed by the combination of gene variants that creates an optimal environment for drug sensitivity or resistance. We hypothesize that the interaction of pathways generated by functional HR or *BRCA* proteins and disrupted *TP53* protein creates the ideal molecular setting for Vigil therapy responsiveness. Mutation in TP53 is likely an early oncogenic event and likely results in clonal cell *TP53* neoantigen expression. Coupled with proficient homologous recombination of *BRCA1/2* wild-type gene patients may achieve low TMB, but also low intratumor heterogeneity (ITH). While low TMB potentially results in decreased CD4+/CD8+ T cells infiltration, this signaling pattern and its associated low ITH may provide for more effective and consolidated T cell response towards clonal neoantigens.

In conclusion, despite sample size limitation, we demonstrate proof of support for the use of DNA analytical methods to separate resistant and sensitive populations to Vigil. These techniques create a robust approach to analyze how the nodal network relationship between genes affects clinical response to Vigil when used as maintenance therapy in advanced stage III/IV resectable disease patients. These results are hypothesis generating and warrant further investigation. Moreover, these results further support novel use of network-based analysis to identify other more sensitive gene targets and potentially additional novel targeted therapeutic combinations with Vigil and possibly other immunotherapeutics.

Chapter 6

Conclusion

In this project, we investigate the practical application of the PB model on selected proteins that play significant roles in the spread, treatment, and prevention of COVID-19 virus diseases. We utilized our recent research to develop fast algorithms and high-performance computing techniques. To this end, we solved the boundary integral form of the PB equation on the molecular surfaces of these proteins. These calculations produce both the electrostatic solvation energy as a global measurement and the electrostatic surface potential for local details of the selected proteins. We investigated the parallel performance of two competing solvers for the boundary integral PB equations on these selected proteins. By considering the advantages of current algorithms and computer hardware, we focused on the parallelization of the TABI solver using MPI on CPUs and the DSBI solver using KOKKOS on GPUs. Our numerical simulations show that the DSBI solver on one A-100 GPU is faster than the TABI solver with MPI on 64 CPUs when the number of elements is smaller than 250,000.

When both GPU and MPI are available and the triangulation quality is good enough so that the TABI preconditioner is not needed for GMRES convergence, we recommend that the GPU-accelerated DSBI solver be used when the number of boundary elements is below 250,000. Otherwise, the MPI-based TABI should be used. If the number of elements becomes so large such that the memory on a CPU task cannot hold an entire tree, we recommend consideration of a domain-decomposition MPI scheme [15, 51, 85, 86]. We note that the memory usage for TABI scales linearly with problem size. When one million boundary elements are used, the memory usage is a little bit over 1GB. Thus for popular tasks on clusters with at least 64G memory per MPI rank, we can handle problems as large as approximately 64 million boundary elements, which is sufficient for simulating middle-large proteins with up to tens of thousands atoms. For even larger biomolecules, e.g. the viral capsids of Zika or H1N1 virus with up to tens of millions atoms [55], domain decomposition approach can be considered [51].

Additionally, we developed scale-free and uniform electrostatic features both accurately and efficiently. They were able to accurately predict solvation energy on their own and in combination with topological features described in [65]. We incorporate these physics-informed features in topology *and* electrostatics and demonstrate that both methods merged together improve model performance, which suggests a benefit of our approach. We described the dependence on the order p in the multipole expansion to the evaluation metrics. This motivates the use of different types of features for better performance of biological quantities relevant to drug discovery/design.

In this project, we provide a deep-learning neural network (DNN) based biophysics model to predict protein properties. The model uses multi-scale and uniform topological and electrostatic features generated with protein structural information and force field, which governs the molecular mechanics. The topological features are generated using the element-specified persistent homology (ESPH) method while the electrostatic features are fast computed using a Cartesian treecode. These features are uniform in number for proteins with various sizes thus the broadly available protein structure database can be used in training the network. These features are also multi-scale thus the resolution and computational cost can be balanced by the users. The machine learning results on over 4000 protein structures show the efficiency and fidelity of these features in representing the protein structure and force field for the prediction of their biophysical properties such as electrostatic solvation energy. Tests on topological or electrostatic features alone and the combination of both showed the optimal performance when both features are used. This model shows its potential as a general tool in assisting biophysical properties and function prediction for broad biomolecules using data from both theoretical computing and experiments.

We have developed methods to link the connections between protein structure and function that help overcome limitations associated with complex and large proteins. We designed mathematical models that can be solved with efficient and accurate numerical algorithms to ultimately discover the useful information that is hidden in complex protein structural data.

Chapter 7
Software Dissemination

The code for all parallel solvers are available on GitHub. The MPI code can be found on https://github.com/elyssasliheet/tabi_mpi_code. The GPU code can be found on https://github.com/yangxinsharon/bimpb-parallelization.

The code to generate electrostatic and topological features and labels as well as all machine learning models can be found on Elyssa Sliheet's personal github page: https://github.com/elyssasliheet.

For cancer research in collaboration with Gradalis Inc, data can be shared following an approved request for a specific research question. Requests may be declined by Gradalis, Inc if deemed to pose a conflict of interest or competitive risk.

BIBLIOGRAPHY

[1] X. Yang, E. Sliheet, R. Iriye, D. Reynolds and W. Geng, *Optimized parallelization of boundary integral poisson-boltzmann solvers*, *Computer Physics Communications* **299** (2024) 109125. vii

[2] E. Sliheet, M. Robinson, S. Morand, K. Choucair, D. Willoughby, L. Stanbery et al., *Network based analysis identifies tp53m-brca1/2wt-homologous recombination proficient (hrp) population with enhanced susceptibility to vigil immunotherapy*, *Cancer Gene Therapy* (2022) . viii

[3] http://www.rcsb.org/pdb/home/home.do. 2, 6, 8, 42, 74

[4] V. Cherezov, D. M. Rosenbaum, M. A. Hanson, S. G. F. Rasmussen, F. S. Thian, T. S. Kobilka et al., *High-Resolution Crystal Structure of an Engineered Human beta2-Adrenergic G Protein–Coupled Receptor*, *Science* **318** (2007) 1258–1265, [`http://science.sciencemag.org/content/318/5854/1258.full.pdf`]. 2

[5] F. Dong, M. Vijaykumar and H. X. Zhou, *Comparison of calculation and experiment implicates significant electrostatic contributions to the binding stability of Barnase and Barstar*, *Biophys. J.* **85** (2003) 49–60. 2

[6] N. Huang, Y. Chelliah, Y. Shan, C. A. Taylor, S.-H. Yoo, C. Partch et al., *Crystal structure of the heterodimeric CLOCK:BMAL1 transcriptional activator complex*, *Science* **337** (2012) 189–194, [`http://www.sciencemag.org/content/337/6091/189.full.pdf`]. 2

[7] D. A. Beard and T. Schlick, *Modeling salt-mediated electrostatics of macromolecules: the discrete surface charge optimization algorithm and its application to the nucleosome*, *Biopolymers* **58** (2001) 106–115. 2

[8] Y. C. Zhou, B. Lu and A. A. Gorfe, *Continuum electromechanical modeling of protein-membrane interactions*, *Phys. Rev. E* **82** (Oct, 2010) 041923. 2

[9] K. M. Callenberg, O. P. Choudhary, G. L. de Forest, D. W. Gohara, N. A. Baker and M. Grabe, *Apbsmem: A graphical interface for electrostatic calculations at the membrane*, *PLoS ONE* **5** (09, 2010) 1–12. 2

[10] D. D. Nguyen, B. Wang and G.-W. Wei, *Accurate, robust, and reliable calculations of Poisson-Boltzmann binding energies*, *J. Comput. Chem.* **38** (2017) 941–948. 2

[11] L. Wilson, J. Hu, J. Chen, R. Krasny and W. Geng, *Computing electrostatic binding energy with the TABI Poisson–Boltzmann solver*, *Communications in Information and Systems* **22** (2022) 247–273. 2, 21

[12] T. Simonson, G. Archontis and M. Karplus, *Free energy simulations come of age: Protein-ligand recognition*, *Acc. Chem. Res.* **35** (2002) 430–437, [http://dx.doi.org/10.1021/ar010030m]. 2

[13] J. A. Wagoner and N. A. Baker, *Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms*, *Proceedings of the National Academy of Sciences* **103** (2006) 8331–8336, [http://www.pnas.org/content/103/22/8331.full.pdf]. 2

[14] N. Unwin, *Refined structure of the nicotinic acetylcholine receptor at 4Å resolution*, *J. Mol. Biol.* **346** (2005) 967 – 989. 2

[15] J. Chen, W. Geng and D. Reynolds, *Cyclically paralleled treecode for fast computing electrostatic interactions on molecular surfaces*, *Comput. Phys. Commun.* **260** (2021) 107742. 2, 21, 96

[16] J. Chen and W. Geng, *On preconditioning the treecode-accelerated boundary integral (TABI) Poisson-Boltzmann solver*, *J. Comput. Phys.* **373** (2018) 750–762. 3, 21, 27, 28

[17] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre and A. Jemal, *Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries*, *CA: a cancer journal for clinicians* **68** (2018) 394–424. 3

[18] S. Coburn, F. Bray, M. Sherman and B. Trabert, *International patterns and trends in ovarian cancer incidence, overall and by histologic subtype*, *International journal of cancer* **140** (2017) 2451–2460. 3

[19] Z. Momenimovahed, A. Tiznobaik, S. Taheri and H. Salehiniya, *Ovarian cancer in the world: epidemiology and risk factors*, *International journal of women's health* (2019) 287–299. 3

[20] A. Yoneda, M. E. Lendorf, J. R. Couchman and H. A. Multhaupt, *Breast and ovarian cancers: a survey and possible roles for the cell surface heparan sulfate proteoglycans*, *Journal of Histochemistry & Cytochemistry* **60** (2012) 9–21. 3

[21] T. Walsh, S. Casadei, M. K. Lee, C. C. Pennil, A. S. Nord, A. M. Thornton et al., *Mutations in 12 genes for inherited ovarian, fallopian tube, and peritoneal carcinoma identified by massively parallel sequencing*, *Proceedings of the National Academy of Sciences* **108** (2011) 18032–18037. 3

[22] A. Toss, C. Tomasello, E. Razzaboni, G. Contu, G. Grandi, A. Cagnacci et al., *Hereditary ovarian cancer: not only brca 1 and 2 genes*, *BioMed research international* **2015** (2015) . 3

[23] A. J. Cortez, P. Tudrej, K. A. Kujawa and K. M. Lisowska, *Advances in ovarian cancer therapy*, *Cancer chemotherapy and pharmacology* **81** (2018) 17–38. 3

[24] D. Jelovac and D. K. Armstrong, *Recent progress in the diagnosis and treatment of ovarian cancer*, *CA: a cancer journal for clinicians* **61** (2011) 183–203. 3

[25] O. W. Foley, J. A. Rauh-Hain and M. G. Del Carmen, *Recurrent epithelial ovarian cancer: an update on treatment*, *Oncology* **27** (2013) 288. 3

[26] R. P. Rocconi, B. J. Monk, A. Walter, T. J. Herzog, E. Galanis, L. Manning et al., *Gemogenovatucel-t (vigil) immunotherapy demonstrates clinical benefit in homologous recombination proficient (hrp) ovarian cancer*, *Gynecologic oncology* **161** (2021) 676–680. 4, 81

[27] N. McGranahan, A. J. Furness, R. Rosenthal, S. Ramskov, R. Lyngaa, S. K. Saini et al., *Clonal neoantigens elicit t cell immunoreactivity and sensitivity to immune checkpoint blockade*, *Science* **351** (2016) 1463–1469. 4

[28] https://pdb2pqr.readthedocs.io/en/latest/, "pdb2pqr package documentation." 6

[29] http://openbabel.org/index.html, "Open babel chemistry toolbox." 6

[30] P. L. Kastritis and A. M. J. J. Bonvin, *On the binding affinity of macromolecular interactions: daring to ask why proteins interact*, *J R Soc Interface 10:20120835.* (2012) . 7

[31] J. P. Vandervaart, N. L. Inniss, T. Ling-Hu, G. Minasov, G. Wiersum, M. Rosas-Lemus et al., *Serodominant SARS-CoV-2 Nucleocapsid Peptides Map to Unstructured Protein Regions*, *Microbiology Spectrum* **11** (2023) e00324–23. 8

[32] D. C. Dinesh, D. Chalupska, J. Silhan, E. Koutna, R. Nencka, V. Veverka et al., *Structural basis of RNA recognition by the SARS-CoV-2 nucleocapsid phosphoprotein*, *PLOS Pathogens* **16** (12, 2020) 1–16. 9

[33] S. Kang, M. Yang, S. He, Y. Wang, X. Chen, Y.-Q. Chen et al., *A SARS-CoV-2 antibody curbs viral nucleocapsid protein-induced complement hyperactivation*, *Nature Communications* **12** (2021) 2697. 9

[34] J. D. Jackson, *Classical Electrodynamics*. john Wiley & Sons, Inc., 1999. 12, 17

[35] M. J. Holst, *The Poisson-Boltzmann Equation: Analysis and Multilevel Numerical Solution*. PhD thesis, UIUC, 1994. 16

[36] W. Geng, *A boundary integral Poisson–Boltzmann solvers package for solvated bimolecular simulations*, *Computational and Mathematical Biophysics* **3** (2015) 43–58. 16, 17

[37] W. Geng and G. W. Wei, *Multiscale molecular dynamics using the matched interface and boundary method*, *J Comput. Phys.* **230** (2011) 435–457. 16

[38] J. G. Kirkwood, *Theory of solution of molecules containing widely separated charges with special application to zwitterions*, *J. Comput. Phys.* **7** (1934) 351 – 361. 17

[39] W. Cai, Z. Xu and A. Baumketner, *A new FFT-based algorithm to compute Born radii in the generalized Born theory of biomolecule solvation*, *Journal of Computational Physics* **227** (2008) 10162–10177. 19

[40] R. J. Zauhar and R. S. Morgan, *The rigorous computation of the molecular electric potential*, *J. Comput. Chem.* **9** (1988) 171–187. 20

[41] A. Juffer, B. E., B. van Keulen, A. van der Ploeg and H. Berendsen, *The electric potential of a macromolecule in a solvent: a fundamental approach*, *J. Comput. Phys.* **97** (1991) 144–171. 20, 22, 23, 24

[42] J. Liang and S. Subranmaniam, *Computation of molecular electrostatics with boundary element methods*, *Biophys. J.* **73** (1997) 1830–1841. 20

[43] A. H. Boschitsch, M. O. Fenley and H.-X. Zhou, *Fast boundary element method for the linear Poisson-Boltzmann equation*, *J. Phys. Chem. B* **106** (2002) 2741–2754, [`http://dx.doi.org/10.1021/jp013607q`]. 20, 21

[44] A. J. Bordner and G. A. Huber, *Boundary element solution of the linear Poisson-Boltzmann equation and a multipole method for the rapid calculation of forces on macromolecules in solution*, *J. Comput. Chem.* **24** (2003) 353–367. 20

[45] B. Lu, X. Cheng and J. A. McCammon, *A new-version-fast-multipole-method-accelerated electrostatic calculations in biomolecular systems*, *J. Comput. Phys.* **226** (2007) 1348 – 1366. 20, 21, 25

[46] J. Barnes and P. Hut, *A hierarchical O(NlogN) force-calculation algorithm*, *Nature* **324** (12, 1986) 446–449. 21, 25, 26

[47] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, *J. Comput. Phys.* **73** (1987) 325 – 348. 21

[48] L. F. Greengard and J. Huang, *A new version of the fast multipole method for screened coulomb interactions in three dimensions*, *J. Comput. Phys.* **180** (2002) 642 – 658. 21

[49] B. Lu, X. Cheng, J. Huang and J. A. McCammon, *Order N algorithm for computation of electrostatic interactions in biomolecular systems*, *Proceedings of the National Academy of Sciences* **103** (2006) 19314–19319, [`http://www.pnas.org/content/103/51/19314.full.pdf+html`]. 21

[50] W. Geng and F. Jacob, *A GPU-accelerated direct-sum boundary integral Poisson-Boltzmann solver*, *Comput. Phys. Commun.* **184** (2013) 1490 – 1496. 21

[51] L. Wilson, N. Vaughn and R. Krasny, *A GPU-accelerated fast multipole method based on barycentric Lagrange interpolation and dual tree traversal*, *Computer Physics Communications* **265** (2021) 108017. 21, 25, 96, 97

[52] P. Li, H. Johnston and R. Krasny, *A Cartesian treecode for screened Coulomb interactions*, *J. Comput. Phys.* **228** (2009) 3858–3868. xii, 21, 25, 26, 36

[53] W. Geng and R. Krasny, *A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules*, *J. Comput. Phys.* **247** (2013) 62 – 78. 21, 22, 24, 25, 33

[54] B. Lu and J. A. McCammon, *Improved boundary element methods for Poisson-Boltzmann electrostatic potential and force calculations*, *J. Chem. Theory Comput.* **3** (2007) 1134–1142, [`http://dx.doi.org/10.1021/ct700001x`]. 24

[55] L. Wilson, W. Geng and R. Krasny, *TABI-PB 2.0: An Improved Version of the Treecode-Accelerated Boundary Integral Poisson-Boltzmann Solver*, *The Journal of Physical Chemistry B* **126** (09, 2022) 7104–7113. 24, 97

[56] W. Geng, *Parallel higher-order boundary integral electrostatics computation on molecular surfaces with curved triangulation*, *J. Comput. Phys.* **241** (2013) 253 – 265. 24

[57] J. Chen, J. Tausch and W. Geng, *A Cartesian FMM-accelerated Galerkin boundary integral Poisson-Boltzmann solver*, *Journal of Computational Physics* **478** (2023) 111981. 24, 25

[58] Y. Saad and M. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, *SIAM J. Sci. Stat. Comput.* **7** (1986) 856–869, [http://dx.doi.org/10.1137/0907058]. 25

[59] Z.-H. Duan and R. Krasny, *An adaptive treecode for computing nonbonded potential energy in classical molecular systems*, *J. Comput. Chem.* **22** (2001) 184–195. 25

[60] K. Lindsay and R. Krasny, *A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow*, *J. Comput. Phys.* **172** (2001) 879 – 907. 25

[61] J. Chen, W. Geng and G.-W. Wei, *MLIMC: Machine learning-based implicit-solvent Monte Carlo*, *Chinese Journal of Chemical Physics* **34** (2021) 683–694, [https://doi.org/10.1063/1674-0068/cjcp2109150]. 29, 42

[62] P. Ren and J. W. Ponder, *Polarizable atomic multipole water model for molecular mechanics simulation*, *J. Phys. Chem. B* **107** (2003) 5933–5947, [http://dx.doi.org/10.1021/jp027815+]. 35

[63] Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder et al., *Polarizable atomic multipole-based amoeba force field for proteins*, *J. Chem. Theory Comput.* **9** (2013) 4046–4063, [http://dx.doi.org/10.1021/ct4003702]. 35

[64] J. Tausch, *The fast multipole method for arbitary Green's functions*, *Contemporary Mathematics* **329** (2003) 307–314. xii, 35, 36

[65] Z. Cang and G.-W. Wei, *Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions*, *PLOS Computational Biology* **13** (07, 2017) 1–27. 37, 38, 39, 40, 44, 56, 57, 76, 97

[66] R. Ghrist, *Barcodes: the persistent topology of data*, *Bulletin of the American Mathematical Society* **45** (2008) 61–75. 39

[67] D. A. Perlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. Debolt et al., *AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules*, *Comp. Phys. Commun.* **91** (1995) 1–41. 42

[68] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. States, S. Swaminathan and M. Karplus, *CHARMM: A program for macromolecular energy, minimization, and dynamics calculations*, *J. Comput. Chem.* **4** (1983) 187–217. 42

[69] C. Zhang, C. Lu, Z. Jing, C. Wu, J.-P. Piquemal, J. W. Ponder et al., *Amoeba polarizable atomic multipole force field for nucleic acids*, *J. Chem. Theory Comput.* **14** (04, 2018) 2084–2108. 42

[70] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li et al., *Comparative assessment of scoring functions: The casf-2016 update*, *Journal of Chemical Information and Modeling* **59** (02, 2019) 895–913. 42

[71] S. Pahari, L. Sun and E. Alexov, *PKAD: a database of experimentally measured pKa values of ionizable groups in proteins*, *Database: Journal of Biological Databases and Curation* **article ID baz024** (Jan, 2019) 1–7. 42

[72] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins and D. D. Cox, *Hyperopt: a python library for model selection and hyperparameter optimization*, . 44

[73] M. F. Sanner, A. J. Olson and J. C. Spehner, *REDUCED SURFACE: An efficient way to compute molecular surfaces*, *Biopolymers* **38** (1996) 305–320. xvi, 53

[74] W. Humphrey, A. Dalke and K. Schulten, *VMD – visual molecular dynamics*, *J. Mol. Graphics* **14** (1996) 33–38. xiii, 55

[75] M. J. Landrum, J. M. Lee, M. Benson, G. R. Brown, C. Chao, S. Chitipiralla et al., *Clinvar: improving access to variant interpretations and supporting evidence*, *Nucleic acids research* **46** (2018) D1062–D1067. 81

[76] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas et al., *String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets*, *Nucleic acids research* **47** (2019) D607–D613. 82

[77] B.-Q. Li, T. Huang, L. Liu, Y.-D. Cai and K.-C. Chou, *Identification of colorectal cancer related genes with mrmr and shortest path in protein-protein interaction network*, *PloS one* **7** (2012) e33393. 82

[78] C. Liu, J. Zhao, W. Lu, Y. Dai, J. Hockings, Y. Zhou et al., *Individualized genetic network analysis reveals new therapeutic vulnerabilities in 6,700 cancer genomes*, *PLoS computational biology* **16** (2020) e1007701. 83, 84, 93, 94

[79] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage et al., *Cytoscape: a software environment for integrated models of biomolecular interaction networks*, *Genome research* **13** (2003) 2498–2504. 85

[80] C. Von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini et al., *String: known and predicted protein–protein associations, integrated and transferred across organisms*, *Nucleic acids research* **33** (2005) D433–D437. 85

[81] Y. Li, X. Zhou, J. Liu, Y. Yin, X. Yuan, R. Yang et al., *Differentially expressed genes and key molecules of brca1/2-mutant breast cancer: evidence from bioinformatics analyses*, *PeerJ* **8** (2020) e8403. 89

[82] Z. Xu, Y. Zhou, Y. Cao, T. L. A. Dinh, J. Wan and M. Zhao, *Identification of candidate biomarkers and analysis of prognostic values in ovarian cancer by integrated bioinformatics analysis*, *Medical oncology* **33** (2016) 1–8. 94

[83] D. Hanahan and R. A. Weinberg, *Hallmarks of cancer: the next generation*, *cell* **144** (2011) 646–674. 94

[84] D. Hanahan and R. A. Weinberg, *The hallmarks of cancer*, *cell* **100** (2000) 57–70. 94

[85] N. Vaughn, L. Wilson and R. Krasny, *A GPU-accelerated barycentric Lagrange treecode*, in *2020 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, pp. 701–710, 2020. 96

[86] J. K. Salmon, M. S. Warren and G. S. Winckelmans, *Fast parallel tree codes for gravitational and fluid dynamical n-body problems*, *Int. J. Supercomputer Appl* **8** (1986) 129–142. 96