Force Field Parameterization in Molecular Simulation by Machine Learning Methods

Gong Chen

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

LJLL

Sorbonne Université
LJLL

Doctoral School **École Doctorale Sciences Mathématiques de Paris Centre**

University Department **Laboratoire Jacques-Louis Lions**

Thesis defended by **Gong Chen**

Defended on **October 19, 2023**

In order to become Doctor from Sorbonne Université

Academic Field **Mathématiques appliquées**

Speciality **Analyse numérique**

Force Field Parameterization in Molecular Simulation by Machine Learning Methods

Thesis supervised by Yvon Maday

Committee members

Referees	Gábor Csányi	Professor at University of Cambridge	
	Gabriel Stoltz	Professor at Ecole des Ponts ParisTech	
Examiners	Bruno Després	Professor at Sorbonne Université	Committee President
	Virginie Ehrlacher	Professor at Ecole des Ponts ParisTech	
	Mihai-Cosmin Marinica	Research Director at CEA Saclay	
	Jean-Philip Piquemal	Professor at Sorbonne Université	
Guest	Bertrand Mercier	Professor at CEA/INSTN	
Supervisor	Yvon Maday	Professor at Sorbonne Université	

Colophon

Doctoral dissertation entitled "Force Field Parameterization in Molecular Simulation by Machine Learning Methods", written by Gong Chen, completed on November 2, 2023, typeset with the document preparation system LaTeX and the yathesis class dedicated to theses prepared in France.

This thesis has been prepared at

Laboratoire Jacques-Louis Lions
Sorbonne Université
Campus Pierre et Marie Curie
4 place Jussieu
75005 Paris
France

☎ +33 1 44 27 42 98
Web Site `https://ljll.math.upmc.fr/`

For my parents and my family.

Force Field Parameterization in Molecular Simulation by Machine Learning Methods

Abstract

Molecular dynamics simulation allows for predictions of material properties, aiding in understanding, researching, and development of new materials. However, the accuracy of molecular force fields has been a long-standing limitation. Traditional force fields assign parameters based on discrete atom types, encoding all information about the atom's chemical environment. Higher accuracy requires more atom types, resulting in a proliferation of redundant parameters and low transferability. In this thesis, we introduce the Graph-Based Force Fields (GB-FFs) model, which employs graph neural networks to process directed molecular graphs and extract continuous atomic representations. These representations are then used to derive a set of force field parameters. GB-FFs model directly learn from quantum chemical energies and forces. In Chapter 1, we initially employ machine learning techniques to predict parameters for polynomial interpolation, demonstrating the effectiveness of neural networks in handling simple parameterization tasks. In Chapter 2, we represent molecules as strings and apply Natural Language Processing models to extract molecular fingerprints. In Chapter 3, we introduce Directed Graph Attention neTworks (D-GATs) to process directed molecular graphs, extracting molecular fingerprints and predicting molecular properties. In the final chapter, building upon the models from previous chapters, we propose the GB-FFs model, which achieves end-to-end molecular force field parameterization. Our methods have proven to be a reliable approach for optimizing and accelerating molecular force field parameterization. Currently, GB-FFs model has been tested and validated on the General AMBER Force Field (GAFF). Furthermore, our model's versatility allows for easy extension and application to other force fields in future research.

Keywords: graph neural networks, molecular force fields, molecular representation learning

Résumé

La simulation de dynamique moléculaire permet de prédire les propriétés des matériaux, contribuant ainsi à la compréhension, la recherche et le développement de nouveaux matériaux. Cependant, la précision des champs de force moléculaire a été une limitation depuis longtemps. Les champs de force traditionnels attribuent des paramètres selon des combinaisons de types d'atomes discrets, ce qui encode toutes les informations sur l'environnement chimique de l'atome. Une plus grande précision nécessite davantage de types d'atomes, ce qui entraîne une prolifération de paramètres redondants et une faible transférabilité. Dans cette thèse, nous présentons le modèle de Graph-Based Force Fields (GB-FFs), qui utilise des réseaux neuronaux de graphes pour traiter des graphes moléculaires dirigés et extraire des représentations atomiques. Ces représentations sont ensuite utilisées pour prédire tous les paramètres des champs de force. Le modèle GB-FFs apprend directement à partir des énergies et des forces. Dans le chapitre 1, nous utilisons initialement l'apprentissage automatique pour prédire les paramètres d'interpolation polynomiale, démontrant que les réseaux neuronaux peuvent réussir des tâches de paramétrisation. Dans le chapitre 2, nous représentons les molécules sous forme de chaînes et utilisons des modèles de traitement du langage naturel pour extraire des empreintes moléculaires. Dans le chapitre 3, nous introduisons Directed Graph Attention neTworks (D-GATs) pour traiter des graphes moléculaires dirigés, extraire des empreintes moléculaires et prédire des propriétés moléculaires. Dans le dernier chapitre, nous proposons le modèle GB-FFs, qui permet la paramétrisation des champs de force de bout en bout. Nos méthodes ont été démontrées comme une approche fiable pour optimiser et accélérer la paramétrisation des champs de force. Actuellement, le modèle GB-FFs a été testé et validé sur le General AMBER Force Field (GAFF). De plus, il pourra facilement être étendu et appliqué à d'autres champs de force également.

Mots clés : réseaux neuronaux graphiques, champs de forces moléculaires, apprentissage de représentation moléculaire

Laboratoire Jacques-Louis Lions
Sorbonne Université – Campus Pierre et Marie Curie – 4 place Jussieu – 75005 Paris – France

Acknowledgement

My doctoral journey commenced at Sorbonne Université in 2019. Over the past four years, I have studied and conducted researches in the areas of my interest. This journey has truly been a valuable and enjoyable experience for me.

First and foremost, I would like to express my deep gratitude to my thesis advisor, Professor Yvon Maday, at Laboratoire Jacques-Louis Lions (LJLL). Professor Maday, your inexhaustible mathematical imagination and enthusiasm have consistently motivated me to go further. Even I came from a different background, you graciously granted and supported me to study and explore the machine learning and molecular dynamics. You proposed the initial concept for this work, and without your generosity, patience, and guidance, none of this would have been possible.

A huge thank also goes to Professor Gabriel Stoltz at Ecole des Ponts. Your invaluable review of my thesis and your constructive corrections significantly improved my manuscript. Your continued involvement as a member of my mid-term committee, offering insights and support, has been instrumental for my Ph.D. study.

I extend my heartfelt appreciation to Professor Gábor Csányi from the University of Cambridge. Despite an complicated schedule, you accepted the role of reviewer for this thesis and participated in my defense. Your expertise in the field of machine learning potentials is truly remarkable, and I am deeply grateful for all the support you have provided.

I offer the most sincere and deepest thanks to Professor Jean-Philip Piquemal from the Laboratoire de Chimie Théorique (LCT) of Sorbonne Université. Your expertise in computational chemistry and molecular force fields have been crucial in guiding my research to its current direction.

Professor Bruno Després of LJLL, I am deeply grateful for your participation in my defense committee and mid-term committee. Your advice has been instrumental in the organization of my work and the progress of this thesis.

I am indebted to Professor Virginie Ehrlacher (Galland) at Ecole des Ponts and Dr. Mihai-Cosmin Marinica from CEA Saclay. Thank you for agreeing to be members of the jury. Your feedback and guidance are invaluable to me.

Beyond the members of my defense committee mentioned above, I would like to express my gratitude to Professor Bertran Mercier. It was your recommendation letter that gave me the opportunity to do an internship and subsequent Ph.D. study with Professor Maday, opening the door to my academic access. My deepest thanks also go to Mr. Jean-Marie Bourgeois-Demersay, dean of Sino-French Institute of Nuclear Engineering and Technology (IFCEN). His genuine care for every student is greatly appreciated. I also want to express my gratitude to Professor Chunyu Zhang, who provided invaluable guidance throughout my undergraduate and master's thesis research, teaching me practical knowledge in numerical simulation methods.

I extend my gratitude to the Extreme Scale Mathematically based Computational Chemistry (EMC2) project team and the weekly meetings organized by Mi-Song Dupuy. A special acknowledgment is reserved for Théo Jaffrelot Inizan, my collaborator from LCT, who helped me

in conducting numerous molecular dynamics simulations. Your insights offered a more robust chemical interpretation to my models, and I promise to be more meticulous in naming models in the future ┗(⁻∇⁻)┛. I wish to thank Thomas Plé for proposing the electron transfer model. I also appreciate the contributions of Louis Lagardère, Olivier Adjoua, and Pier Paolo Poier in developing algorithms and maintaining Tinker-HP, which significantly facilitated my work.

Within my laboratory, there are many people I would like to thank for their support. I extend my gratitude to the secretaries, including Malika Larcher, Salima Merbouha Lounici, Erika Loyson, Corentin Maday, and Catherine Drouet, for their administrative assistance at LJLL. I also acknowledge Khashayar Dadras for providing technical support and maintaining the laboratory servers. My thanks also go to Corentin Lacombe for his assistance during the submission phase of my thesis.

Among my friends in office 15-25 302, I express my appreciation to Jules Pertinand, Anne-Françoise de Guerny, Anouk Nicolopoulos, Valentin Pagès, Houssam Houssein, Elena Ambrogi, Thomas Borsoni, María Cabrera Calvo, Ludovic Souetre, and everyone else who shared our office space. To me, you are more than colleagues. The atmosphere in our office has always been relaxed and friendly. Additionally, within our laboratory, I wish to thank Muhammad Hassan, Olivier Graf, Elise Grosjean, Lise Maurin, Willy Haik, Fatima E. Jabiri, Nicolas Torres, Ramon Oliver Bonafoux, Anatole Guérin, Lucas Journel, Pierre Le Bris, Nicolaï Gouraud, Agustin Somacal, Chourouk El Hassanieh, Nga Nguyen, Dang Toai Phan, Yvonne Bronsard Alama, Matthieu Dolbeault, Roxane Delville Atchekzai, Ioanna-Maria Lygatsika, Antoine Leblond, Charles Elbar, Rémi Robin, Juliette Dubois, Robin Roussel, Cristobal Loyola, Eleanor Gemida, as well as other Ph.D. students and postdoctoral researchers. We have spent pleasant times together at LJLL.

For my fellow Chinese students at Sorbonne Université (Yipeng Wang, Mingyue Zhang, Qingyou He, Rui Li, Liangying Chen, Zeyu Lyu, Boxi Zhao, Jingrui Niu, Ruiyang Dai, Haibo Liu, Zhe Chen, Ruikang Liang, and Siguang Qi), I would like to express my thanks for your assistance as well as your accompany. And I hope you can publish more articles in the future. To those who have left LJLL, including Helin Gong, Allen Juntao Fang, Po-Yi Wu, Siyuan Ma, Xinran Ruan, Shijie Dong, Yangyang Cao, Shengquan Xiang, Liudi Lu, and Jingyi Fu, I wish you continue to achieve greatness in your academic careers.

Very special thanks to all my IFCEN friends in France: Yang Pei, Jinjiang Cui, Chufa Qiu, Kunyu Wang, Linkai Wei, Yizheng Wang, Xueying Wang, Jingyi Wang, and Chujun Lin. You have all achieved the most significant degree in your lives. I will always cherish the happy hours we spent in France. I also extend my appreciation to Dan Liu and Meng Luo, and I wish you all the best.

Finally, I would like to express my heartfelt gratitude to my parents and my family, who have consistently supported me in my research journey.

Contents

Chapter 0

Introduction

0.1 Problem Statement and Motivation

Molecular Force Fields (FFs) have played a crucial role in computational chemistry for decades, serving as empirical models that describe atomic-level interactions and molecular behaviors to facilitate simulations and predictions of various chemical processes. These models date back to pioneering efforts in the mid-20th century to accurately model simple molecules.

FFs comprise distinct terms that account for diverse interactions among atoms within molecules and between molecules in a system. These terms can be typically categorized into three types:

Bonding Terms: These terms describe interactions between atoms within a molecule, taking into account the covalent bonds that involve the sharing of electrons to form electron pairs between atoms. Bond stretching, angle bending, dihedral terms and cross terms are typical examples.

Non-bonding Terms: Non-bonding terms describe interactions among atoms not directly connected by covalent bonds. This category encompasses Van der Waals (VdW) interactions and electrostatic interactions. VdW interactions explain attractive forces among molecules or atoms, while electrostatic interactions account for interactions among charged atoms. These terms are vital for capturing long-range interactions and intermolecular forces within a molecular system.

Polarization Terms: Polarization terms involve assigning partial charges to atoms based on their electronegativity and molecular geometry. These effects are crucial for systems in high-dielectric medium such as water.

Traditional non-polarizable FFs have made significant contributions to computational chemistry but have inherent limitations due to their empirical nature. FF parameters are typically derived from experimental data and quantum mechanical calculations, offering valuable insights but struggling to capture the full complexity of molecular systems. These challenges include:

1. **Limited Atom Types:** Conventional methods assign discrete atom types to represent chemical environments. Non-bonded parameters are assigned according to the atom types. Bond stretching, angle bending, and dihedral configuration parameters are subsequently determined by consulting parameter table, that contain atom type combinations and their corresponding bonding terms. To achieve higher accuracy, atom types must be sufficiently

complex to encode all essential chemical information about the atoms, which poses challenges in fitting all bonding terms.

2. **Parameterization Challenges:** Parameterizing a FF involves determining optimal values for numerous parameters that govern atomic interactions. This procedure is typically time-consuming, demanding extensive manual optimization and often relying on trial-and-error methodologies. Furthermore, the accuracy of FFs heavily depends on the selection of reference data and the developer's expertise, introducing subjectivity and potential biases into the process.

3. **Limited Functional Forms:** The functional form determines the mathematical representation and calculation of interactions among atoms and molecules within FFs. Choosing a particular functional form inherently limits the FFs' capacity to precisely capture all potential molecular interactions. Various FFs adopt distinct functional forms, each accompanied by its unique set of assumptions and approximations.

4. **Limited Representation of Quantum Mechanical Effects:** Quantum mechanical phenomena, including quantum tunneling and zero-point energy, play crucial roles in achieving accurate descriptions of chemical reactions and molecular dynamics. Unfortunately, traditional FFs encounter struggle to adequately incorporate these phenomena, resulting in inaccuracies when predicting reaction pathways and thermodynamic properties.

5. **Inadequate Treatment of Electronic Effects:** Traditional FFs primarily emphasize interatomic interactions using fixed parameters, including partial atomic charges. Although this approach simplifies calculations, it tends to neglect electronic effects, such as polarization and charge transfer, which hold significant importance in numerous chemical processes. Consequently, FFs often fail to accurately capture the dynamic nature of molecular systems.

6. **Insufficient Transferability:** The transferability of FF parameters across diverse molecules and environments poses another substantial challenge. FFs are frequently developed for specific chemical systems, diminishing their reliability when employed in novel contexts. This limitation come from the complexity of accurately representing the wide range of molecular interactions and environments.

While traditional FFs have been successful in many applications, their limitations hinder their applicability to complex molecular systems and accurate dynamic processes. To address these limitations, researchers have explored improved FF variants, such as polarizable FFs (see Section 4.1.2) and Machine Learning (ML) Potentials (see Section 4.1.3). Polarizable FFs aim to capture the dynamic nature of molecular systems by considering polarization effects, which is crucial for intermolecular interactions. In contrast, ML Potentials utilize data-driven ML models to provide energies and forces based on molecular geometries, enabling molecular dynamics simulations without involving explicit functional expressions.

Despite these efforts, both polarizable FFs and ML Potentials have introduced significant computational costs. Given the inherent limitations of traditional FFs and the desire for enhanced accuracy, there is a need to explore alternative parameterization methods that build upon traditional FFs while mitigating the aforementioned challenges. To this end, we propose to abandon atom type based parameterization techniques, and instead utilizing Neural Networks (NNs) for direct atomic embedding and parameter assignment.

This approach, distinct from the previously described ML Potentials, offers the potential to overcome most of the previously mentioned limitations. Specifically, traditional FFs often struggle with the definition of atom types and the comprehensive fitting of parameters (as highlighted

in challenges 1 and 2). However, by employing NNs to extract atomic representations, we can discern the chemical environment without relying on predefined atom types, thus significantly improving efficiency and accuracy.

Additionally, our accelerated parameterization process enables us to explore a wider range of functional forms (as mentioned in challenge 3). For instance, bond stretching and angle bending energy are typically described using harmonic functions, not solely due to superior performance, but because they yield satisfactory results within a certain range. With our parameterization approach, manual tuning is no longer necessary, and what used to take several years for the fitting process and testing (validation) can now be accomplished in just a few days. We can design and experiment with dozens of potential functional forms to select the one that offers optimal performance.

Moreover, our approach significantly enhances the transferability of FF parameters across different molecules and environments (as highlighted in challenge 6). Traditional FFs rely on predefined atom types, limiting their adaptability to new chemical systems. In contrast, by utilizing NNs to extract atomic embeddings, we can capture the diverse range of molecular interactions and environments in a more flexible manner. This is a more transferable and generalizable parameterization scheme, reducing the need for developing system-specific FFs.

By leveraging the capabilities of NNs to extract atomic embeddings and derive parameters, we can revolutionize the process of FF parameterization. This approach not only addresses part of the deficiencies of traditional FFs but also offers an efficient and adaptable method for capturing complex interatomic interactions.

It should be noted that this thesis does not propose a new FF. Instead, our focus is on optimizing existing FFs using innovative technologies. This approach allows us to take advantage of expert knowledge and maintain compatibility with current molecular mechanics simulation packages, such as Amber [1–3] and Tinker [4, 5].

0.2 State of The Art

0.2.1 High-Precision Chemistry Database

ML is fundamentally data-driven, and thus high-precision databases play a crucial role in ML for understanding molecular features. These databases provide a comprehensive and accurate representation of molecular properties and interactions, enabling the development of robust ML models in the field of chemistry.

The significance of high-precision databases lies in their ability to capture the intricate nuances of molecular behaviors and properties, which are often challenging to describe through traditional analytical methods. By applying quantum mechanical calculations and accurate computational techniques, these databases offer meticulous insights at molecular structures, energies, forces, vibrational frequencies, and other relevant properties. This detailed understanding of molecules is indispensable for tasks such as drug design, chemical reaction optimization, property prediction, and the exploration of novel materials.

Generating a high-precision database from scratch can indeed be a resource-intensive and time-consuming process, considering the computational resources required and the need for data screening and processing. Fortunately, With the advancement of computational chemistry, we now have access to a large number of high-precision databases that can provide sufficient data for training complex models. This availability has significantly simplified the process of training complex models.

Based on the data they contain, databases can be categorized into three distinct classes:

Molecular Property Databases: These databases do not incorporate spatial information but instead comprise molecules along with their associated properties, such as solubility and toxicity. Given that certain molecular properties necessitate experimental measurement, such databases may encompass only a limited number of molecules, potentially leading to overfitting. Their primary purpose is to serve as training data for models and aid in the screening of target molecules in initial stage.

For example, the Toxicology in the 21st Century (Tox21) database [6] stands as a widely used collection of chemical compounds and their corresponding biological activities. It primarily supports toxicity prediction and assessment in drug discovery and environmental research. Tox21 consists of a large collection of chemical compounds, which have been screened by high-throughput screening methods against a panel of human cell-based assays to assess their potential effects on various biological targets associated with different toxicological endpoints. These endpoints include nuclear receptor signaling, stress response pathways, and cytotoxicity, among others. The database supplies information on the chemical structures of the compounds, along with their corresponding bioactivity measurements.

Molecular Conformation Databases: In this class, databases are generated through quantum chemical calculations and include atomic position information. They typically encompass data like molecular potential energy, atomic forces, and atomic charges. These databases are utilized for purposes such as fitting FFs, studying intermolecular interactions, and aiding in drug design.

An example of this class is ANI-1 [7, 8], the largest database of Density Functional Theory (DFT) computations for small organic molecules. ANI-1 comprises over 20 million off-equilibrium conformations of 57,462 small organic molecules, extracted from the GDB database [9, 10]. These conformations result from exhaustive sampling of a subset of the GDB-11 database, focusing on molecules containing 1 to 8 heavy atoms and considering only H, C, N, and O species. Electronic and structure calculations are conducted using the ωB97x [11] density functional and the 6–31 G(d) basis set [12], making it valuable for assessing the precision of ML-driven FF parameterization [11].

Other Databases: The third class encompasses databases designed for specific purposes, like the Reaction SMILES dataset [13], utilized to study molecular reactions and predict reaction equations. These databases satisfy specialized research needs and applications beyond the scope of the previous two classes.

In this work, aside from the ANI-1 database, SPICE (Small-Molecule/Protein Interaction Chemical Energies) database [14] also plays a central role. SPICE is primarily designed for simulating interactions between drug-like small molecules and proteins, covering a wide chemical space with 15 elements (H, C, N, O, F, P, S, Cl, Br, I, Li, Na, Mg, K, Ca). It contains over one million conformations, and corresponding energies and forces, making it a valuable molecular conformation database. The computations in SPICE are carried out using the ωB97M-D3(BJ) functional [15, 16] and def2-TZVPPD basis set [17, 18], which are even more accurate than those used in the ANI-1 database. SPICE includes various subsets, each designed to provide specific types of information, including dipeptides, solvated amino acids, PubChem data [19], monomer and dimer information [20], and ion pairs.

With existing high-precision databases, the approach of fitting FFs using ML becomes a viable alternative. By fitting the molecular potential energy, we ensure that FFs perform well across a broad range of molecules. Additionally, fitting atomic forces allows for improvements in FFs' local performance. The combination of these aspects ensures that FF parameterization through ML can yield outstanding results.

0.2.2 Machine Learning and Force Fields

ML has indeed found numerous applications in the field of FFs, revolutionizing various aspects of computational chemistry. Notable contributions of ML in this context include:

Potential Energy Surface Exploration: ML Potentials [21–25] assist in exploring the potential energy surfaces in complex molecular systems. By leveraging the learned patterns from known molecular structures and their corresponding energies, ML models can predict the potential energy landscapes of new molecules. This capability is invaluable for molecular dynamics simulations and the study of chemical reactions.

Force Field Parameterization: ML techniques can be used to analyze large datasets of molecular properties, such as quantum mechanical calculations or experimental data, to refine FF parameters [26–28]. This approach leads to more accurate predictions of molecular behavior and properties.

Property Prediction: Property Prediction: ML models can predict diverse molecular properties, including geometry [29], electronic structure [22], and spectroscopic properties [30]. These predictions aid in understanding molecular behavior, optimizing molecular structures for specific purposes, and accelerating the discovery of materials with desired properties.

Molecule Design: ML algorithms, in conjunction with FFs, support the inverse design of molecules with desired properties [29, 31, 32]. By training models on known relationships between molecular structures and properties, ML models generate new molecules that meet criteria such as stability, reactivity, or targeted interactions. This plays a crucial role in drug design and material design.

There are already some ML models to be applied in FFs. If we use ML to simulate energy and forces directly from molecular geometry, it can be thought as a replacement to FFs. However, if we use ML to optimize FF parameters and speed up the simulation, it should be thought as a complementary tool to FFs.

The classical FFs have developed for more than four decades, resulting in well-optimized molecular mechanics simulation codes. Our ML model provides a set of optimized parameters, allowing us to take advantage of the existing molecular mechanics packages.

0.2.3 Molecular Processing Model

Molecular representation learning has long been crucial in drug discovery and materials science [33–35]. Recent advancements in Natural Language Processing (NLP) and Graph Neural Networks (GNNs) have significantly contribution to this field. NLP treats molecules as one-dimensional sequential tokens, while GNNs view them as two-dimensional topology graphs. GNNs, with their diverse message passing algorithms, exhibit varying performance in detecting chemical environments and predicting molecular properties.

Although text representations are unnatural for molecules, they offer several advantages over GNNs. Text-based representations benefit from comprehensive ML frameworks designed for text processing, benefiting from the strong connection between NLP and sequence modeling. Furthermore, training generative models is often simpler with text, as generating valid text is less complex than generating valid graphs. Consequently, sequence models are commonly employed for generative and unsupervised learning of chemical space. In contrast, GNNs tend to excel in supervised learning tasks and can effectively incorporate spatial features [36, 37].

Natural Language Processing

NLP techniques have found applications in various aspects of molecule processing, enabling researchers to analyze and understand chemical information more effectively. Here are some applications of NLP in molecule processing:

Chemical Named Entity Recognition (NER): It is the task of identifying and classifying chemical entities within text, such as chemical names, formulas, and identifiers [38, 39]. NLP models can automatically extract chemical information from scientific literature, patents, or other textual sources. This aids in building chemical databases, completing chemical knowledge, and facilitating information retrieval.

Chemical Reactions: Transformers, as detailed in Section 2.2, have demonstrated remarkable effectiveness in discovering the insights in chemical reactions and processes. Schwaller et al. demonstrated their effectiveness in synthetic pathway analysis, predicting products from reactants and reagents [40]. These models can also be fine-tuned to predict synthetic yield [40, 41]. Schwaller et al. have also trained a transformer to classify reactions into organic reaction classes, yielding an intriguing map of chemical reactions [42].

Chemical Text Generation: NLP models can create chemical text, including molecular structures, names, and descriptions. This capability supports tasks like generating compound names, proposing chemical reactions [40], and aiding chemical synthesis planning [41]. Generated text promote exploration in chemical design spaces and aids in database creation.

Chemical Property Prediction: Combining NLP with ML models enables the prediction of molecular properties from textual descriptions [43–45]. By training on chemical text and experimental data, it becomes possible to predict solubility, toxicity, bioactivity, and other descriptors. This aids in virtual screening, drug discovery, and property optimization.

These applications underscore NLP's crucial role in molecule processing techniques, enabling the efficient extraction, analysis, and generation of chemical insights from textual data. Researchers benefit from accelerated exploration and deeper understanding of chemical compounds, reactions, and processes.

Graph Neural Networks

Molecules are naturally represented as graphs, where atoms as nodes and bonds as edges. This approach encodes both atom and bond features into embeddings, preserving structural information. In some models, geometry information is also encoded into the atom features, further enriching the information.

GNNs are designed to effectively process graph-structured data. There are mainly two types of models: Graph Convolutional Networks (GCNs) [46–48] and Graph Attention Networks (GATs) [49, 50]. GCNs extend the convolutional operation to graphs, enabling information propagation between connected nodes and capturing structural information. GATs leverage self-attention mechanisms (refer to Section 2.2.2) to assign attention weights to each node's neighbors dynamically. This allows GATs to focus on the most relevant nodes during information aggregation and capture more fine-grained relationships within the graph. Due to their reliance on attention mechanisms, GATs show more flexibility in algorithm design. Thus we primarily consider GATs in our work.

Message Passing Neural Network (MPNN). MPNN [33] provides a powerful framework of GNNs. MPNN follows a two-step process: message passing and readout. In the message passing step, each node aggregates information from its neighboring nodes. This allows the nodes to

update their own states, capturing both local and global structural information. After multiple rounds of message passing, the readout step aggregates the updated node states to generate a fixed-size graph-level representation. This representation can then be used for various downstream tasks, such as molecular property prediction and chemical reaction prediction. During the message passing phase, hidden states h_i^t at node i in t-th interaction layer are updated based on edge states e_{ij} (between node i and j) and messages m_i^{t+1} according to:

$$m_i^{t+1} = \sum_{j \in \mathcal{N}(i)} M^t(h_i^t, h_j^t, e_{ij}) \tag{1}$$

$$h_i^{t+1} = U^t(h_i^t, m_i^{t+1}) \tag{2}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i in the graph $\mathcal{G}$. M^t is message function and U^t is vertex update function.

Directed Message Passing Neural Network (D-MPNN). Traditional models treat molecular graphs as undirected graphs. In [51, 52], Directed MPNN (D-MPNN) have proposed directed edges to avoid loops during the message passing phase of the algorithm (refer to Figure 1).

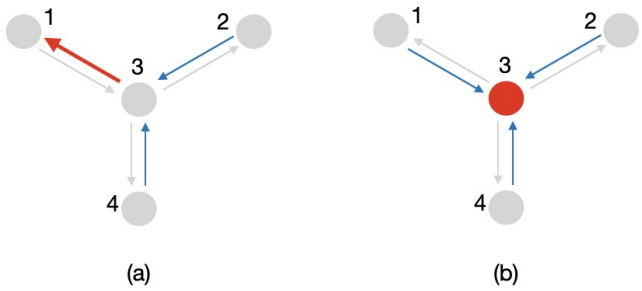

Figure 1: **Example of message passing in D-MPNN: (a) Update of edge states:** The edge $3 \rightarrow 1$ is updated by (edge $2 \rightarrow 3$ and edge $4 \rightarrow 3$) **(b) Update of node states:** The node 3 is updated by (edge $1 \rightarrow 3$, edge $2 \rightarrow 3$ and edge $4 \rightarrow 3$)

D-MPNN follows the message passing functions in Equations (1) and (2). The edge states $\{h_{ij}^{t+1}\}$ at layer $t+1$ for all connected nodes i and j are updated by:

$$m_{ij}^{t+1} = \sum_{k \in \mathcal{N}(i) \setminus j} M^t(h_i^t, h_k^t, h_{ki}^t) = \sum_{k \in \mathcal{N}(i) \setminus j} h_{ki}^t \tag{3}$$

$$h_{ij}^{t+1} = U^t(h_{ij}^t, m_{ij}^{t+1}) = \sigma(h_{ij}^0 + W_m \cdot m_{ij}^{t+1}) \tag{4}$$

where $\sigma(\cdot)$ is the `ReLU` activation function, $W_m \in \mathbb{R}^{D_h \times D_h}$ and D_h is the dimension of model.

The node states $\{h_1, h_2, ..., h_N\}$ are not updated. Instead, they are derived from the initial node features F^n and the edge hidden states at last layer T that direct to the node:

$$h_i = \sigma(W_a \cdot [F_i^n, \sum_{k \in \mathcal{N}(i)} h_{ki}^T]) \tag{5}$$

where $W_a \in \mathbb{R}^{D_h \times 2D_h}$ and $[.,.]$ is the concatenation operation.

D-MPNN has only applied the simple aggregate functions, which limits models' performance. The following models apply attention mechanism [49] to improve aggregate functions.

Directed Graph Attention Neural Network (DGANN). This model [53] uses attention mechanism to update bond states and atom states (see Figure 2(a)). However, DGANN first updates the directed bonds and only the outputs at last layer would be used to update atom states and molecule-level representations. In contrast, our model Directed Graph Attention neTworks (D-GATs), the bond/atom/molecular states are updated in each interaction layer, ensuring tighter coupling.

Graph Edge Attention (GEA). In X. Han's work [54], GEA (see Figure 2(b)) is based on additive attention mechanism [55], using addition rather than multiplication to compute attention weights. However, additive attention has been found to be less efficient than dot-product attention [49]. GEA has explored various ReadOut functions, including max-pooling, sum-pooling, and set2set [56]. In contrast, our D-GATs model leverages a more robust structure called supervirtual node.

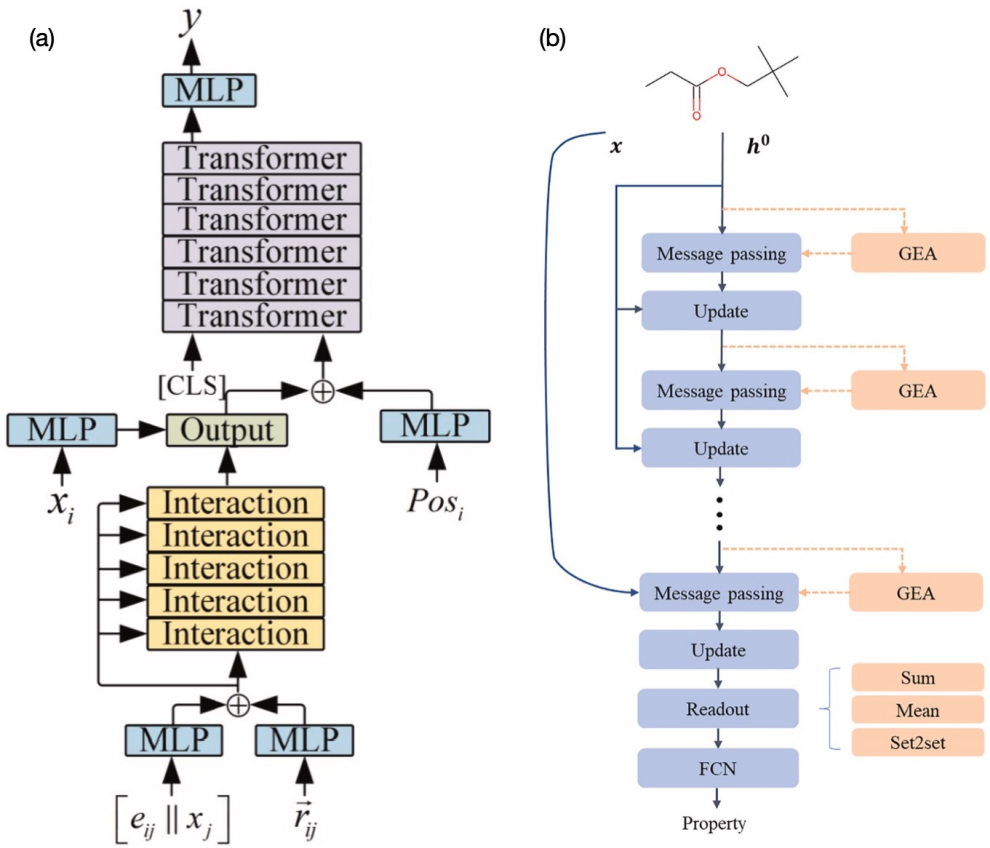

Figure 2: **DGANN and GEA: (a)Framework of DGANN (from Figure 1(b) in [53]).** The yellow layers are to update bond states and only the final bond states (green box) will be used to update atom states (in purple boxes) **(b)Framework of GEA (from Figure 5 in [54]).** GEA also applies attention mechanism to update bond/atom states. But the attention to each part is optional. And `ReadOut` function can be Sum/Mean/Set2set.

0.2.4 ESPALOMA

This model (called extensible surrogate potential optimized by message-passing, or ESPALOMA) [28] is the newest work on constructing end-to-end optimizable FFs with continuous atom embeddings. It consists of three stages (see Figure 3):

- Stage 1: Using GNNs to perceive chemical environments and update atom embeddings.
- Stage 2: Constructing continuous bond, angle and torsion embeddings by pooling to preserve appropriate symmetries.
- Stage 3: Computing FF parameters from atom, bond, angle, and torsion embeddings by feed-forward NNs.

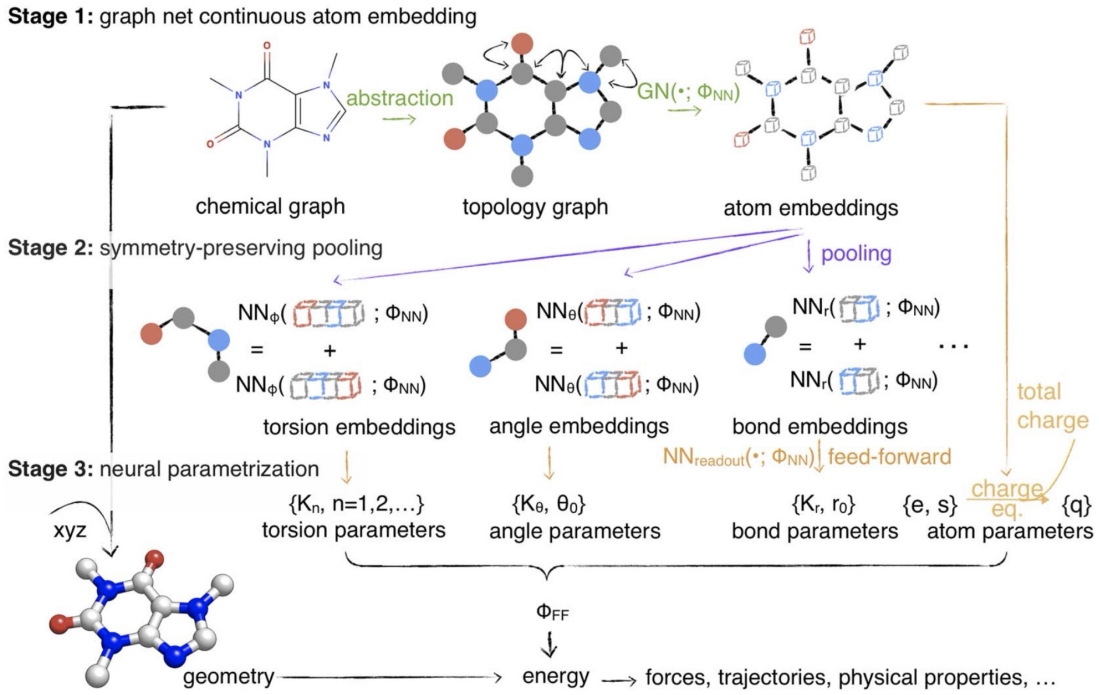

Figure 3: **Framework of ESPALOMA** (from Fig. 1 in [28])

The traditional schemes assign discrete atom types based on attributes of atoms and their neighbors, and human chemical intuition is used to assign specific parameters for different combinations of these types. In the stage 1, to compute continuous atom embeddings, ESPALOMA applies three 128-units GraphSAGE [57] layers with ReLU activation function to update the atom features in each layer.

In stage 2, certain symmetries exist in molecular mechanics potentials, where terms in the potential function remain unchanged when the involved atoms are permuted. Inspired by Janossy pooling [58], ESPALOMA enumerates the relevant equivalent atom permutations. For instance, bond terms exhibit symmetry when the atoms in the bond are exchanged. The bond embedding h_r for connected atoms v_i and v_j are computed by the atom embeddings h_v (Equation (2) in [28]):

$$h_{r_{ij}} = NN_r([h_{v_i}, h_{v_j}]) + NN_r([h_{v_j}, h_{v_i}]) \tag{6}$$

where $[.,.]$ denotes concatenation and NN_r are the NNs for bonds. In this way, $h_{r_{ij}} = h_{r_{ji}}$.

In stage 3, ESPALOMA uses feed-forward NNs to convert continuous embeddings into FF parameters and applies Charge equilibration [59] to predict charge distribution.

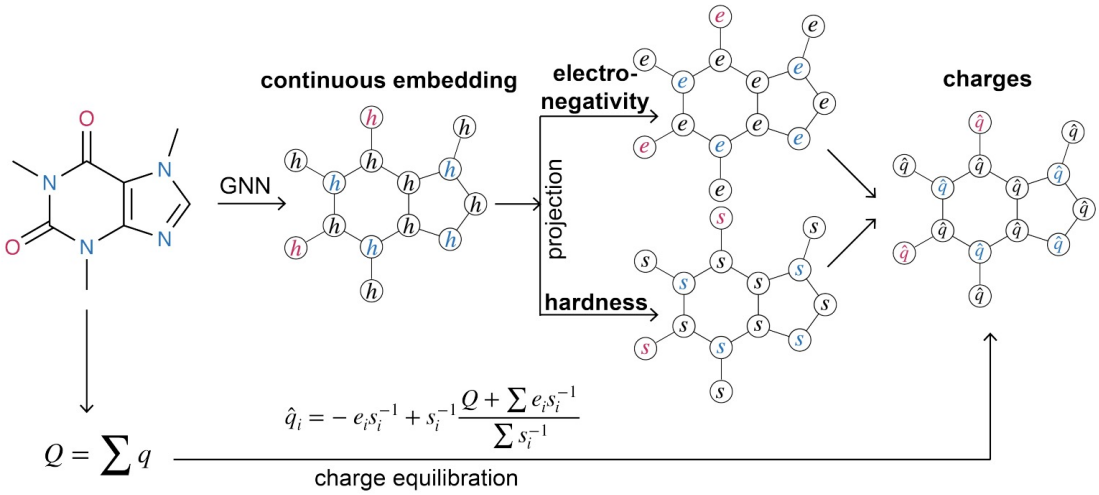

Figure 4: **Framework of ESPALOMA charge** (from Figure 1. in [60]). e_i and s_i are the first-order and second-order derivative of the potential energy in charge for each atom. Q is the total charges and $\hat{q}_i$ is partial atomic charge at atom i.

Our Graph-Based Force Fields (GB-FFs) model shares the same logic as ESPALOMA, utilizing GNNs to extract atom embeddings to parameterize the FFs. However, there are several points where our approach differs:

- GB-FFs model utilizes fully attention-based D-GATs to extract atomic embeddings. Compared to ESPALOMA, which uses GraphSAGE [57], our approach has demonstrated superior predictive power in modeling molecular properties.

- Unlike Janossy pooling [58], which explores all possible combinations, GB-FFs' parameter generator separates the inputs embedding according to inherent symmetry, thereby reducing the computational cost by half.

- GB-FFs model does not employ charge equilibration. Instead, it directly predicts the charge transfer between connected atoms based on directed bonds. This approach is easier to interpret, aligning better with human intuition.

- GB-FFs aims to re-parameterize and optimize existing FFs, primarily trained on ANI-1 and SPICE datasets. In contrast, ESPALOMA starts from basic intra-molecular interactions and requires a larger dataset to build models from scratch. Therefore, ESPALOMA operates independently of existing force fields.

0.3 Layout of the Thesis

0.3.1 Chapter 1: A Preliminary Research on Polynomial Interpolation

Main Idea: ML operates as a data-driven approach, learning from sampled examples, which can include various data types, ranging from text and images to molecular properties and function values. Its primary goal is to develop the capability to evaluate objects beyond the training samples. This process is analogous to fitting an interpolating function and then performing interpolation or extrapolation.

Based on the analogy of interpolation functions, we can infer at least two possible applications of ML. One approach is completely data-driven. ML model evaluates the objects outside the sampling space through learning from existing sample data [61–65]. For example, by learning the change of function $y = x^2$ on the interval $[-1, 1]$, we expect ML to predict the function values on the interval $[1, 2]$. Another approach combine both data-driven and physics-based elements, forming a hybrid scheme. In this context, ML only provides key information, such as coefficients or parameters. By integrated with existing knowledge, it is capable to evaluate the objects beyond the sampling space [28, 62, 66]. For instance, with the same function $y = x^2$ on $[-1, 1]$, we can derive a set of coefficients for polynomial interpolation. These coefficients, along with the appropriate basis functions, construct an approximate function for predicting function values within the range $[1, 2]$.

In summary, the choice between these two approaches depends on whether the ML output necessitates supplementary information or expert knowledge. Chapters 2 and 3, where we employ NNs for molecular property prediction, align with the first type. Similarly, ML Potentials for predicting potential energy and forces also fall into this category. However, the core objective of this thesis, re-parameterizing the FFs via ML, involves parameters that must be coupled with functions to compute potential energy. Therefore, this task should be of the second type.

Utilizing NNs to predict the coefficients of polynomial interpolation serves as a validation of ML's applicability to tasks of the second type. The goals of this chapter are:

- Gain proficiency in ML algorithms and deep learning models.
- Evaluate the robustness of NNs in handling noise by applying them to polynomial fitting.
- Investigate the potential of NNs in parameterization tasks.

Model Architecture: Similar to the definition of Lebesgue constant, we define the Lebesgue constant with perturbation $\Lambda_N^{\eta,p}$ as:

$$\Lambda_N^{\eta,p} := \max_{f,\varepsilon} \frac{\|P_N(f+\varepsilon) - P_N(f)\|_{L^p}}{\|\varepsilon\|_{L^\infty}} \tag{7}$$

where N is the degree of polynomials, P_N is the approximation function, $p \in \{2, \infty\}$, $f \in L^p$ is the function to approximate, ε is the noise, $\frac{\|\varepsilon\|_{L^\infty}}{\|f\|_{L^\infty}} \leq \eta$, with η the level of noise.

We feed the function values into the NNs to obtain the coefficients α for polynomial interpolation. Then we use these coefficients along with the corresponding basis functions $\{\ell_i\}$ to construct an approximate function $G_N(f) = \sum_i \alpha_i \ell_i$.

We have considered several combinations (Chebyshev/Equidistant points, Lagrange/Legendre polynomials, level of noise in training data) under two scenarios: one-dimensional (1D) and two-dimensional (2D) cases. For 1D scenario (refet to Section 1.2), we utilize the feed-forward NNs with fully connected layers. For 2D case (refer to Section 1.3), we employ the convolutional NNs [67–69] combined with global max pooling [70, 71].

Main Conclusions: Generally speaking, NNs can effectively preserve the essential properties of original functions and fit them with polynomials. Despite being inherently nonlinear, G_N maintains remarkable additivity when applied to polynomial interpolation (Section 1.2.4).

Our results indicate that NNs' performance is closely related to the training data. The introduction of noise during training stage aids models in discovering and filtering noise. Compared to classical interpolation, G_N yields smaller $\Lambda_N^{\eta,p}$ (Section 1.2.6).

Throughout this chapter, we've validated the potential of NNs for parameterization tasks. However, the accuracy of G_N is not comparable to classical interpolation when the degree of polynomials is high (Section 1.2.3). We infer that NNs may not be suitable for low-dimensional parameterization tasks. Additionally, the generalizability of G_N is somewhat constrained—it cannot be readily applied to fit functions outside the families present in the training data (Supplementary information in Section 1.5).

0.3.2 Chapter 2: Molecule Processed as Text

Main Idea: NLP allows a computer to be capable of understanding and interpreting the contents of documents, including the contextual nuances of the language within them. Simplified Molecular-Input Line-Entry system (SMILES) [72] is proposed to represent molecules in a simple way (Section 2.1). It is a line notation which represents the chemical structures in a graph-based definition, where the atoms, bonds and rings are encoded in a graph and represented in text sequences. SMILES is considered as a "chemical language" and its strings follow a regular grammar. For example, the molecule Melatonin, shown in Figure 5, is expressed as: "CC(=O)NCCC1=CNc2c1cc(OC)cc2". SMILES is more like a real language we use in daily life for the following reasons:

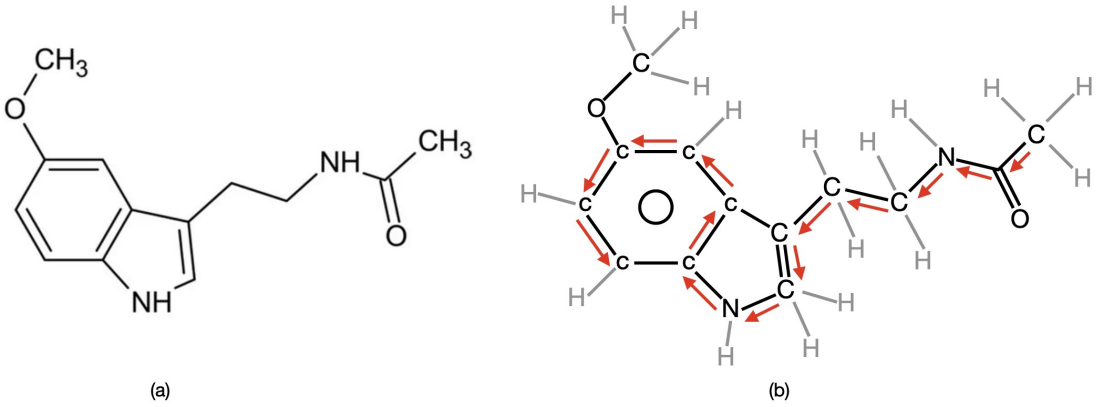

Figure 5: **Example of SMILES: (a)** The molecule Melatonin. **(b)** Melatonin expressed in SMILES: CC(=O)NCCC1=CNc2c1cc(OC)cc2 (all hydrogen atoms are ignored). The order of atoms in the main chain is indicated with red arrows.

- SMILES/Sentences are both variable length sequences with strict grammar.
- SMILES/Sentences are composed of characters/words.
- The order of characters/words in SMILES/sentence is important to the outputs. For instance, "John helps Susan" and "Susan helps John" have different meanings. "COC" and "CCO" are two different molecules.

SMILES for one molecule is equivalent to a sentence, with each character, such as in "C=C", corresponding to an atom or a relationship between atoms. Consequently, every character within SMILES can be considered equivalent to a word in a phrase. This is why we employ NLP techniques when working with SMILES.

Model Architecture: The Transformer architecture is initially introduced as an encoder-decoder structure by Vaswani et al. [49]. However, in our application, we exclusively utilize the encoder based on self-attention mechanism to extract molecular fingerprints.

The self-attention mechanism (Section 2.2.2) operates by computing attention weights between all character pairs in input SMILES sequence. These weights signify the significance of each character in relation to others, enabling the model to assign higher importance to relevant information. Consequently, the Transformer encoder efficiently captures dependencies across the entire sequence, regardless of the distance between positions. This attention mechanism also empowers the model to handle variable-length sequences without relying on recurrent connections or convolutional filters.

For a more detailed view of our model, refer to Figure 6 and Section 2.2.3.

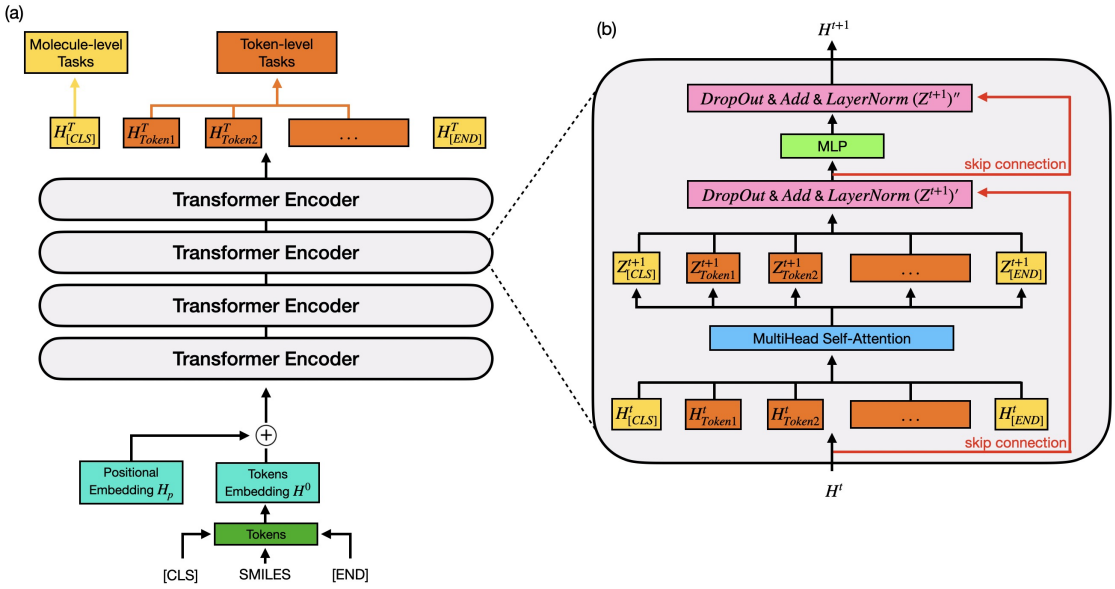

Figure 6: **Framework of our NLP model: (a)** It is made up of 4 Transformer encoder layers with 16 attention heads at each layer. Dimension of model is 512 and dimension of feed-forward networks is 1024. **(b)** Operation in one transformer encoder

Main Conclusions: The experimental results demonstrate the effectiveness of NLP models in predicting molecular properties, suggesting their potentials in drug design and molecule discovery. However, applying NLP models to derive FF parameters is still challenging. SMILES characters belong to sequential data, posing difficulties in understanding molecular geometry information. We have tried to apply NLP to predict FF parameters, but the relative error is sometimes up to 10%, far above the acceptable error.

0.3.3 Chapter 3: Molecule Processed as Directed Graph

Main Idea: Molecules can be naturally represented using molecular graphs, with atoms as nodes and bonds as edges. However, two primary methods for molecular representation, SMILES and molecular graphs, have distinct advantages and disadvantages. SMILES provides a compact representation that is easily stored and transmitted but lacks spatial structural information. In contrast, molecular graphs capture the spatial structure but are more complex to manipulate.

Traditional models typically treat molecular graphs as undirected graphs. Nevertheless, D-MPNN suggests that directed bonds can prevent loops during message passing and reduce information over-mixing (Section 3.2.4). Therefore, we introduce Directed Graph Attention neTworks (D-GATs) [73], which combine the benefits of directed graphs with the efficiency of attention mechanisms, to process directed molecular graphs (Section 3.3).

Model Architecture: In D-GATs, messages are associated with directed bonds rather than atoms to prevent "tottering" [74]. As depicted in Figure 1, the message from node $3 \rightarrow 1$ do not propagate further in subsequent iterations. However, in undirected graphs, the message from node $3 \rightarrow 1$ will be transmitted back to node 3 in, generating an loop in the message passing trajectory [51, 52].

We use RDKit [75] to process SMILES and extract atom/bond features. These features and molecular graph (in Lewis structure [76]) serve as inputs for D-GATs. The update procedure in D-GATs, as depicted in Figure 7(b), follows a specific sequence. In each interaction layer, we employ the attention mechanism three times to independently update directed bond states, atom states, and molecular representations.

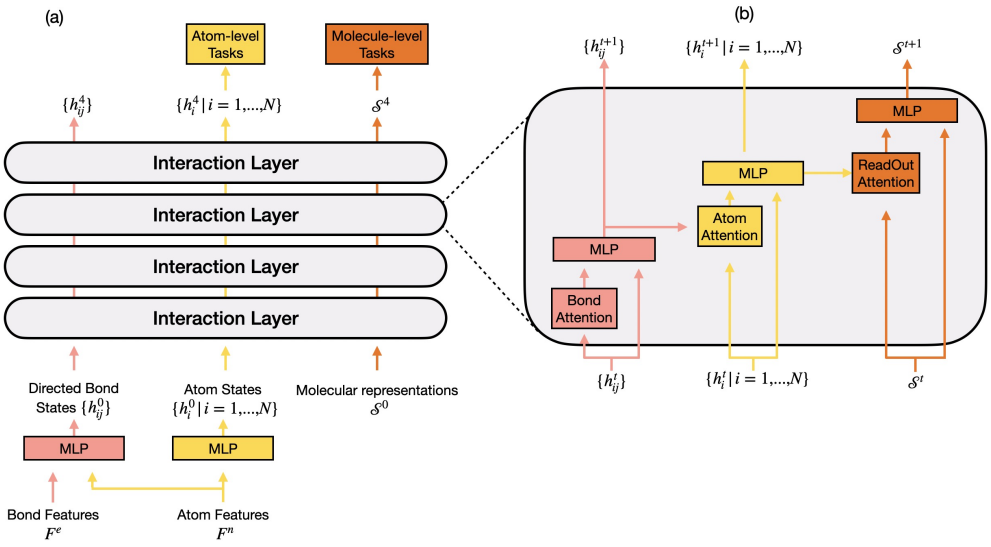

Figure 7: **Framework of D-GATs: (a)** Inputs, 4 interaction layers and outputs. **(b)** Details in each interaction layer

The trainable parameters in MultiLayer Perception (MLP) are:

$$W_1^e, W_2^e, W_1^n, W_2^n, W_1^{\mathcal{S}}, W_2^{\mathcal{S}} \in \mathbb{R}^{D_h \times D_h}$$

with D_h is the dimension of hidden states (or called dimension of model) and $\sigma(\cdot)$ is the ReLU

activation function. The $e, n, \mathcal{S}$ in superscript represent that the corresponding parameters are used for updating bonds, atoms, and molecular representations correspondingly.

- Update of Directed Bond States

The notations in Equation (3-5) are directly token from paper [52] and they are different from those in Equation (8-13). This is because we use e_{ij} to indicate undirected bonds while $\vec{p}(ij)$ indicates directed bonds. Additionally, we use attention mechanism to replace summation as aggregate function, making our functions more complicated. Therefore, we have applied some modifications to notations.

Denote by $\vec{p}(ij)$ the directed bond from atom i to j and by $h^t_{\vec{p}(ij)} \in \mathbb{R}^{D_h}$ bond states in the t-th layer. We define an ensemble $\mathcal{E} = \{\vec{p}(ij)\} \cup \{\vec{p}(ki) | k \in \mathcal{N}(i), k \neq j\}$. Following the framework and notations in [33, 52], we compute the bond messages $m^{t+1}_{\vec{p}(ij)}$ by the equations:

$$m^{t+1}_{\vec{p}(ij)} = M^{t+1}_e(h^t_q | q \in \mathcal{E}) = \sum_{q \in \mathcal{E}} \alpha^{t+1}_{\vec{p}(ij),q}(h^t_q W^{t+1}_{V^e}) \tag{8}$$

where $\mathcal{N}(i)$ denotes the neighbor atoms of atom i and $W^{t+1}_{V^e} \in \mathbb{R}^{D_h \times D_h}$. The attention-based message functions M^{t+1}_e compute the coefficients $\alpha_{\vec{p}(ij),q}$ $(q \in \mathcal{E})$ by attention mechanism.

Next is a MLP where the messages are used to update directed bond states by U^{t+1}_e:

$$h^{t+1}_{\vec{p}(ij)} = U^{t+1}_e(h^t_{\vec{p}(ij)}, m^{t+1}_{\vec{p}(ij)}) = W^e_2(\sigma(W^e_1(\texttt{LayerNorm}(h^t_{\vec{p}(ij)} + m^{t+1}_{\vec{p}(ij)})))) \tag{9}$$

where `LayerNorm`, a type of normalization technique, is from [77].

- Update of Atom States

Note $\{h^t_1, h^t_2, ..., h^t_N\}, h^t_i \in \mathbb{R}^{D_h}, i = 1, ..., N$ as the atom states in t-th layer and N is the number of atoms. Followed by the update of directed bond states, atom messages m^{t+1}_i are updated through vertex message functions M^{t+1}_n:

$$m^{t+1}_i = M^{t+1}_n(h^t_i, h^{t+1}_{\vec{p}(ji)} | j \in \mathcal{N}(i)) = \alpha^{t+1}_{i,i}(h^t_i W^{t+1}_{V^n}) + \sum_{j \in \mathcal{N}(i)} \alpha^{t+1}_{i,j}(h^{t+1}_{\vec{p}(ji)} W^{t+1}_{V^n}) \tag{10}$$

where $W^{t+1}_{V^n} \in \mathbb{R}^{D_h \times D_h}$.

For $j \in \mathcal{N}(i) \cup \{i\}$, the attention weights $\alpha^{t+1}_{\vec{p}(ij),q}$ $(q \in \mathcal{E})$ are computed by attention mechanism. Next is to update atom states in MLP:

$$h^{t+1}_i = U^{t+1}_n(h^t_i, m^{t+1}_i) = W^n_2(\sigma(W^n_1(\texttt{LayerNorm}(h^t_i + m^{t+1}_i)))) \tag{11}$$

- Update of Molecular Representations

Note $\mathcal{S} \in \mathbb{R}^{D_h}$ as the molecular representations in t-th layer. There exists a virtual node connected to all atoms in molecule and it is used as molecule-level representation. Given the updated atom states h^{t+1}_i, the molecular representations $\mathcal{S}^{t+1}$ are updated by `ReadOut` function defined as:

$$m^{t+1} = \texttt{ReadOut}^{t+1}(\mathcal{S}^t, h^{t+1}_i | i = 1, 2, ..., N) = \alpha^{t+1}_{\mathcal{S}}(\mathcal{S}^t W^{t+1}_{V^{\mathcal{S}}}) + \sum_{j=1}^{N} \alpha^{t+1}_i(h^{t+1}_i W^{t+1}_{V^{\mathcal{S}}}) \tag{12}$$

where $W_{V^{\mathcal{S}}}^{t+1} \in \mathbb{R}^{D_h \times D_h}$, $\{\alpha_i^{t+1} | i \in [1, N] \cup \{\mathcal{S}\}$ are also from self-attention mechanism and N is the number of atoms.

Finally, the molecular representations are:

$$\mathcal{S}^{t+1} = U_{\mathcal{S}}^{t+1}(\mathcal{S}^t, m^{t+1}) = W_2^{\mathcal{S}}(\sigma(W_1^{\mathcal{S}}(\texttt{LayerNorm}(\mathcal{S}^t + m^{t+1})))) \tag{13}$$

Main Conclusions: Traditional GNNs treat molecular graphs as undirected graphs. We propose D-GATs that follow the common framework of MPNNs and explore a bond-level message passing algorithm completely relying on scaled dot-product attention mechanism. D-GATs outperform state-of-the-art baselines on 13/15 molecular property prediction tasks, spanning various molecular properties and dataset sizes, on MoleculeNet benchmarks [78].

Our extensive evaluation underscores the superiority of D-GATs' message passing algorithm in learning molecular representations. Notably, D-GATs achieve this using only basic atom as well as bond features, outperforming strong baseline models in both classification and regression tasks. D-GATs consist of three key components: an attention-based scheme for updating bond and atom representations, a `ReadOut` function for extracting molecular representations, and a linear classifier for downstream tasks. This model operates at the complexity of $\mathcal{O}(N)$. These results highlight D-GATs' potential as a powerful tool for molecular property prediction.

It is important to note that D-GATs are tailored for small-sized graphs, ideal for typical molecular properties. Although they are less computationally efficient than undirected graph models (roughly three times slower), the additional computational cost is acceptable. However, the presence of rings in graphs can disrupt directed message flow, To avoid this problem, model depth must be carefully decided. These two limitations make D-GATs particularly suitable for molecular graphs but less suitable for large or dense graphs like those found in social networks.

The code and pre-trained models of D-GATs are publicly available at `https://github.com/GongCHEN-1995/D-GATs`.

0.3.4 Chapter 4: Graph-Based Force Fields Model

Main Idea: In previous chapters, we have successfully applied ML methods to process molecules and predict their properties, primarily focusing on molecular-level properties and treating each molecule as an independent entity. However, FFs in molecular mechanics require a deep understanding of intra-atomic interactions, to evaluate the energy and forces between atoms based on nuclear coordinates [79, 80]. This chapter explores the application of GNNs to FF parameterization and discusses how we optimize legacy FFs by adjusting functions.

Our work focuses on optimizing General AMBER Force Field (GAFF) [81], one of the most widely used classical FFs for simulating organic molecules. GAFF is originally developed by Junmei Wang and Peter A. Kollman in 2004, as an extension of the Amber force field to be compatible with existing versions for proteins and nucleic acids and has parameters for most organic molecules composed of C, N, O, H, S, P, F, Cl, Br and I. In fact, our model can be extended to other legacy FFs without modification in scheme.

GAFF incorporates a comprehensive parameter set, encompassing bond stretching, angle bending, dihedral angles, and non-bonded interactions. These parameters allow for accurate modeling and simulation of organic molecules, even under extreme conditions like high pressure or low temperature. Due to its computational efficiency, relative reliability and especially its simple functional form, GAFF has been widely implemented in most popular molecular simulation software packages.

$$E_{GAFF} = E_{stretching} + E_{bending} + E_{dihedrals} + E_{non-bonded} \tag{14}$$

with

$$
\begin{aligned}
E_{stretching} &= \sum_{bonds} K_r(r - r_{eq})^2 \\
E_{bending} &= \sum_{angles} K_\theta(\theta - \theta_{eq})^2 \\
E_{dihedrals} &= \sum_{n=1}^{4} \frac{V_n}{2}\left[1 + \cos(n\phi - \gamma)\right] + \frac{V_2}{2}\left[1 + \cos(2\varphi - \pi)\right] \\
E_{non-bonded} &= \sum_{i<j}\left[\epsilon_{ij}\left(\frac{\sigma_{ij}^{12}}{R_{ij}^{12}} - 2\frac{\sigma_{ij}^{6}}{R_{ij}^{6}}\right) + \frac{q_i q_j}{\varepsilon R_{ij}}\right]
\end{aligned}
\tag{15}
$$

As shown in Equation (14) and (15), in GAFF, the bond stretching and angle bending interactions are modelled using a harmonic potential, making it non-reactive and thus greatly simplifying the parameterization process. The torsional potential is expressed as a Fourier series. For non-bonded interactions, the VdW interactions are described by a 12-6 Lennard Jones potential [82, 83]. The electrostatic potential, on the other hand, is governed by Coulomb's law.

- r is the bond length. θ is the bond angle. ϕ is the torsional angle. φ is the improper torsional angle. R_{ij} is the distance between non-bonded atoms i and j. These values are determined by the molecular conformations. Knowing the coordinate of each atom, the molecular dynamics package computes these values at high efficiency.

- r_{eq} and θ_{eq} are equilibrium structural parameters. K_r, K_θ, V_n are force constants. n is multiplicity and γ is phase angle for torsional angle parameters. $\gamma = 0$ if n is an odd number. Otherwise $\gamma = \pi$.

- Dihedral terms contain the torsional terms (first terms in $E_{dihedrals}$) and improper torsional terms (second terms in in $E_{dihedrals}$). To distinguish, we note torsional angle as ϕ and note improper torsional angle as φ. As the torsional energy is about the angle, it is expressed as a four-term Fourier series.

- VdW parameters ϵ, σ, and charge q characterize the non-bonded potentials. Partial charges are assigned using a restrained electrostatic potential fit (RESP) model [84, 85]. ϵ, σ follow Lorentz-Berthelot combination rules [86, 87].

Parameterizing a FF is a challenging task since its accuracy and transferability heavily depend on parameter quality. This process is time-consuming, often taking years, and involves empirical heuristics, experimental, and computational data. Additionally, these FFs rely on local frames, known as atom types or atom classes, to assign parameters (e.g., bonds, angles). To enhance the generalization and reliability of FFs, one tendency is to expand the atom type space. However, this leads to an explosion in the number of possible bonding compositions, introducing significant complexity in the parameter fitting process. Moreover, even with modern parameter optimization frameworks [88] and sufficient data, FF parameters defined by fixed atom types can sometimes suffer from low transferability.

While FFs have traditionally been fitted to experimental data and continue to be so, recent advances in computational power and enhanced scalability of *ab-initio* methods have provided new opportunities. Consequently, there has been a growing effort to leverage ML for predicting FF parameters, while still maintaining the predefined functional forms of the potential. Wang

et al. have proposed ESPALOMA [28], a pioneering approach that combine GNNs and automatic differentiation to predict FF parameters. By focusing on intramolecular interactions, they demonstrated that GNNs can effectively predict FF parameters based on potential energies.

We propose the Graph-Based Force Fields (GB-FFs) model as a universal framework for FF parameterization. The GB-FFs model automatically derive accurate FF parameters using only basic atom and bond features. This model offers a continuous alternative to traditional discrete atom typing schemes, eliminating the need for assigning atom types and enabling FF parameters predictions directly from atomic representations. This novel approach extends the generalizability of FFs.

Unlike ESPALOMA, the GB-FFs model does not initiate training from scratch but is designed to re-parameterize and optimize existing FFs.

Model Architecture: The parameter that we want to fit are : $K_r, r_{eq}, K_\theta, \theta_{eq}, V_n, \epsilon_{ij}, \sigma_{ij}, q_i$. We can bring them to the Equation 14 to compute potential energies and forces. The inputs include atom/bond features and Lewis structure.

The full GB-FFs model comprises three key components: a molecule processing model (Section 4.2.2), a symmetry-preserving parameter generator (Section 4.2.3) and a charge transfer model (Section 4.2.4). It is a framework to do molecule-related missions as it shows outstanding ability in detecting chemical environments and returns accurate atomic and bond fingerprints, which can be applied to various tasks.

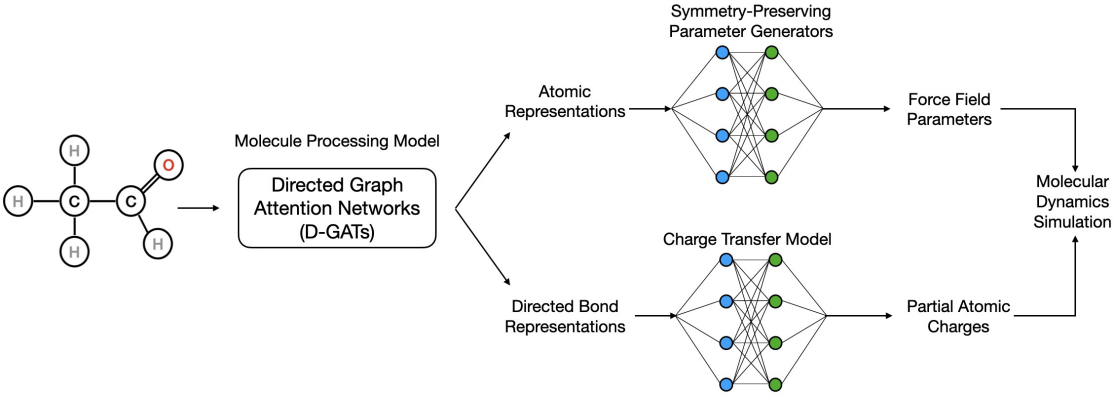

Figure 8: **Framework of Graph-Based Force Fields (GB-FFs) Model.** It consists of a molecule processing model (to accept atom/bond features and Lewis structure and extract embedding), a symmetry-preserving parameter generator (to predict all FF parameters) and a charge transfer model (to predict charge distribution).

Assigning atom type to atoms or deriving FF parameters are typical atom-level tasks. Following the idea of directed bonds introduced in D-MPNNs [52], our model adopts D-GATs (presented in Section 3.3) as its backbone. In comparison to other ML-based molecular processing models, D-GATs show stronger ability in detecting local chemical environments and eliminating unnecessary message flows. Notably, D-GATs have outperformed state-of-the-art baselines on 13/15 molecular property prediction tasks. To enhance the robustness, we use Smooth Maximum Unit (SMU) [89] as activation function because SMU can smoothly approximate the general Maxout [90] family, ReLU, Leaky ReLU or its variants.

To be compatible with GAFF, we consider compounds made of C, N, O, H, S, P, F, Cl, Br and I. RDKit [75] extracts the basic atomic and bond features, which are then fed into the

molecule processing model. The outputs are a set of atomic representations and directed bond representations.

The parameter generators need to ensure atom ordering symmetries. For example, when predicting bond parameters, any exchange of the two input atomic representations should yield invariant predicted parameters. To achieve this, we split the input atom embeddings according to their intrinsic structure and employ linear transformations to ensure symmetry (Section 4.2.3).

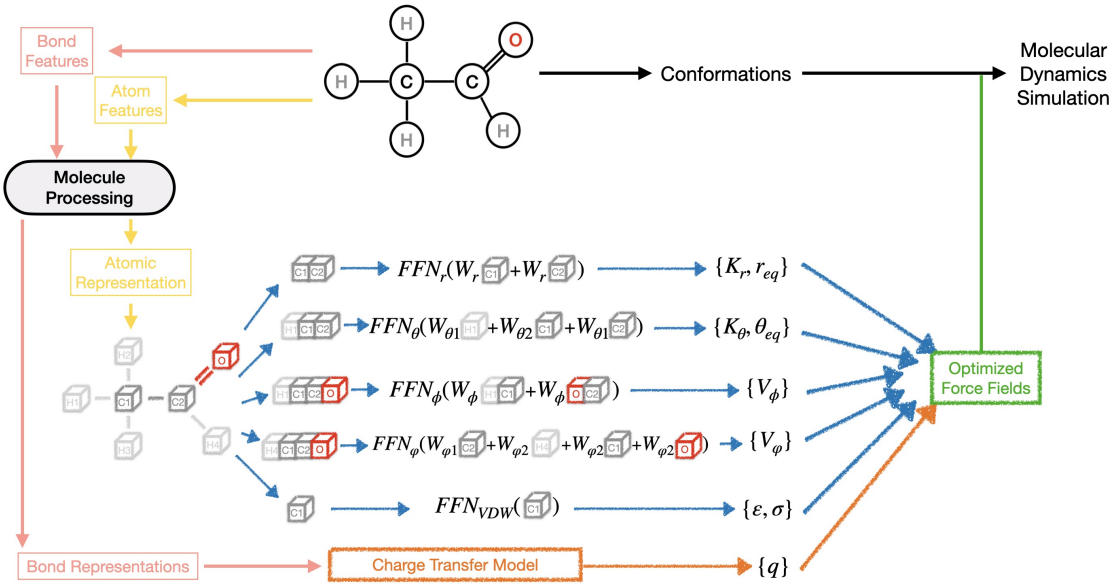

Figure 9: **Symmetry-Preserving Parameter Generators**. For a specified molecule, we input atom and bond features to hierarchical D-GATs and obtain the atomic representations and directed bond representations. The symmetry-preserving parameter generators predict all FF parameters, which can be used to perform molecular dynamics simulation.

To ensure that the net charge of molecule aligns with the actual scenario and to improve the physical meaning of charge distribution, we do not directly predict the charges on each individual atom using atomic expressions. Instead, we utilize the directed bond states from molecule processing model to predict the charge transfer between two connected atoms. As illustrated in Figure 10, the directed bond features are fed into Feed-Forward NNs to determine the charge transfer in the respective bond direction. The final atomic charge is computed by summing the original formal charge and the incoming charges while subtracting the outgoing charges.

We have also made some preliminary attempts in the optimization of FFs functional forms. Firstly, for the stretch energy, we replace the harmonic function with the complete Morse function (Section 4.2.6):

$$E_{bonds} = \sum_{bonds} D(e^{-\alpha(r-r_{eq})} - 1)^2 \tag{16}$$

with $\{D, \alpha, r_{eq}\}$ are the FF parameters and r is the bond length.

Secondly, we add the Urey-Bradley (UB) terms (Section 4.2.7) [91]:

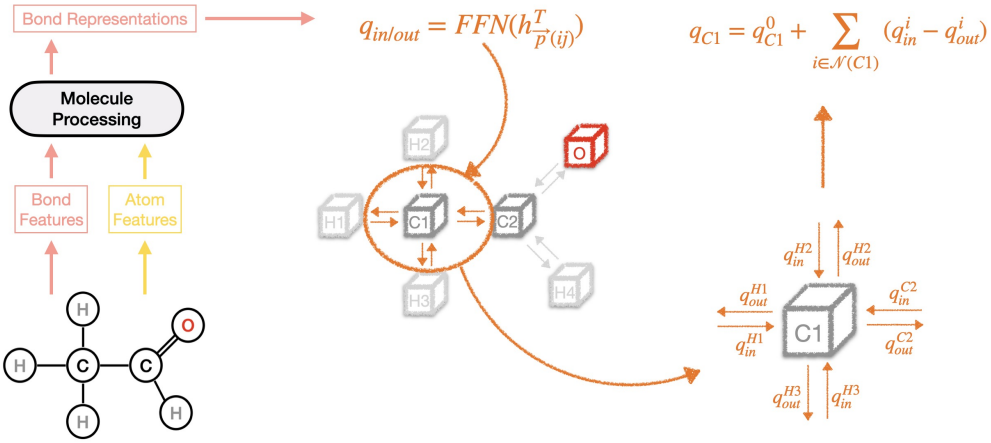

Figure 10: **Charge Transfer Model.** The charge is allowed to transfer between connected atoms and the charge in/out is directly calculated by the directed bond embeddings. The final partial charge of an atom is the original formal charge plus charge flows in and minus the charge flows out.

$$E_{UB} = \sum_{angles} K_{UB}((\frac{r_{UB_{eq}}}{r_{UB}})^2 - 1)^2 \tag{17}$$

with $\{K_{UB}, r_{UB_{eq}}\}$ are the new FF parameters. r_{UB} is the 1-3 endpoints distance.

Main Conclusions: While molecular dynamics simulations are powerful tools for investigating molecular systems, their applicability has long been constrained by the accuracy of the employed FFs. To overcome this limitation and eliminate the reliance on atom types, we have developed the GB-FFs model, operating at the complexity of $\mathcal{O}(N)$, to optimize the FF parameterization.

Our focus is on the optimization of GAFF using D-GATs, to process directed molecular graphs and extract atomic fingerprints. These atom-level representations are then fed into a parameter generator to derive corresponding FF parameters.

Due to the complexity and fragility of FF parameterization, our approach starts with pre-training GB-FFs models on the ANI-1 database [7], followed by fine-tuning on the SPICE [14] and DES370K databases [20]. We extensively validate our GB-FFs model across multiple databases, demonstrating its effectiveness in capturing intermolecular interactions (a reduction in the Root Mean Square Error (RMSE) of intermolecular potential energies from 1.8 to less than 1 kcal/mol on S66×8 database) and energy variations from dihedral angles (on torsion scan database, the RMSE of potential energies drops from 3.5 to about 1.3 Kcal/mol). This ML-based parameterization method set FF parameters free from atom types, relying solely on atomic chemical environments, resulting in GB-FFs model surpassing the original GAFF in terms of both accuracy and generality.

Furthermore, the flexibility of our approach enables its straightforward extension to other non-polarizable FFs. Despite the existence of automatic parameterization procedure, Antechamber [92] is still extremely time-consuming. Our GB-FFs model takes only 0.022 seconds to parametrize a molecule with 96 atoms.

We have also investigated the impact of functional forms on FF performance. On the pre-training ANI-1 database, replacing the harmonic potential with the Morse function to evaluate

stretching energies and introducing Urey-Bradley terms prove to be highly effective in reducing the RMSE in relative potential energy (from 12.7 Kcal/mol to 2.9 Kcal/mol). However, this improvement is less observed when fitting potential energies on SPICE database, though the errors for atomic forces are greatly reduced (from 6.2 to 4.7 Kcal/mol/Å).

In summary, our research addresses some of the limitations of FFs by incorporating ML techniques. Through optimizing the parameterization process with GNNs, we have enhanced FF performance, leading to improved accuracy and efficiency in simulating various molecular systems. These findings open up new possibilities for advancing molecular dynamics simulations and offer a promising method for future researches in this field.

0.4 Conclusions and Perspectives

Machine Learning and Polynomial Interpolation: By replacing classical polynomial interpolation with NNs, we have validated the effectiveness of NNs in performing parametric tasks, demonstrating enhanced stability against noise in our models. However, it is important to acknowledge that, in comparison to classical interpolation methods, NNs may not achieve the same level of accuracy, particularly in low-dimensional scenarios. This is merely a preliminary attempt but it seems like the NNs are not suitable for low dimensional parameterization tasks.

Additionally, in the current setup, we need to retrain a new model whenever we modify the function's dimension, basis functions, or interpolation points. This process is undoubtedly laborious and inefficient. To develop a general NNs interpolation model, it requires a more flexible framework capable of processing various data types and increasing the model's complexity to handle a wider range of functions.

Natural Language Processing and SMILES: While the application of NLP in the context of molecular analysis and property prediction has gained numerous results, it is important to recognize its limitations and explore alternative approaches. The advent of Transformer architectures based on attention mechanisms, have proved effective in extracting molecular fingerprints. These models excel at addressing molecular level tasks, but they may encounter challenges when applied to atomic-level tasks.

Additionally, even though NLP models can be utilized for predicting molecular reactions by treating chemical equations as strings, incorporating existing chemical knowledge and expertise into such process remains challenging. Furthermore, the representation of complex molecules, such as proteins with complex three-dimensional structures, solely using SMILES is insufficient to capture all the chemical intricacies and nuances.

Hence, it becomes evident that while NLP can be applied to in molecular processing, it is not the ultimate solution for all molecular research. A comprehensive approach that considers the unique characteristics and complexities of the molecules being studied is imperative. By combining NLP techniques with other specialized methodologies, we can strive towards more accurate and comprehensive analyses of molecules and their properties."

Directed Graph Attention Networks: Our study focuses on the application of D-GATs for processing directed molecular graphs and predicting molecular properties. Directed molecular graphs offer several advantages over traditional undirected graphs, allowing for the capture of crucial structural information and dynamic relationships within molecules.

D-GATs, as fully attention-based GNNs, provide a flexible framework capable of handling diverse types of molecules. Recent advancements in ML for molecular analysis have witnessed a significant shift towards GNNs as they outperform NLP models in capturing intricate molecular structures. In the case of D-GATs, each interaction layer sequentially updates bond repre-

sentations, atomic representations, and molecular representations, effectively incorporating the chemical context. These representations can be applied to a wide range of tasks involving bonds, atoms, or molecules, showing the adaptability of D-GATs.

The experimental evaluation achieves state-of-the-art results on 13 out of 15 benchmarks, demonstrating the superior performance of D-GATs, and highlighting their potential in molecular processing. These results underscore the remarkable capabilities of D-GATs and their efficacy in various molecular analysis tasks. Moreover, the successful implementation of D-GATs allows for their application across diverse fields such as drug discovery, chemical synthesis planning, and property prediction.

In summary, D-GATs represent a powerful and promising approach to processing directed molecular graphs and predicting molecular properties. Their potential to develop molecular research is evident, promising significant advancements in understanding and manipulating molecules for various applications.

An important future direction for D-GATs is to develop more pre-training strategies to enhance their generalization ability. Another possible direction is to design better message passing algorithm. Currently, our message flows in connected bonds, while higher body order messages [93] could merge information with impressive efficiency, improving model's expressive ability without the need for additional layers.

Force Field Parameterization: Through previous researches, we verified the effectiveness of NNs in performing parameterization tasks, and identified the capability of D-GATs model for handling molecular data. Based on these insights, we have developed a symmetry-preserving parameter generator and a charge transfer model, to derive a comprehensive set of FF parameters directly from bond representations and atom representations.

Considering the fragility of FFs, even slight variations in parameters can lead to unpredictable effects on overall performance. To address this challenge, we initially employ the GAFF as a prototype. Our approach involves training GB-FFs model to reproduce GAFF parameters, and subsequently refining them using potential energy and atomic force data obtained from quantum chemical computations in high accuracy. The well-trained GB-FFs model not only yields improved parameters compared to GAFF but also generates a complementary set of GB-FFs charges.

Experimental results have demonstrated a significant enhancement in the efficiency of our parameterization process. Our GB-FFs model exhibits a speedup of several thousand times compared to the AMBER method. For instance, while AMBER takes 111 seconds to parameterize a 50-atom molecule, our models accomplish this task in a mere 0.018 seconds. Moreover, the model's effectiveness is markedly improved, supported by the reduction in RMSE of the relative potential energy from 6.03 to less than 3.1 Kcal/mol and a decrease in the RMSE of atomic forces from 13.4 to approximately 6 kcal/mol/Å(on SPICE database).

In contrast to traditional approaches, which often involve years of manual parameter fitting, training a GB-FFs model based on existing GAFF parameters takes less than two days. Notably, this process is entirely data-driven, eliminating the need for expert knowledge or manual intervention, resulting in a streamlined and efficient workflow.

Additionally, our research has explored the impact of different functional forms on FFs. The Morse function has demonstrated superior performance over harmonic functions when simulating bond stretching energy over a broader range. Furthermore, the incorporation of Urey-Bradley term into the evaluation of bending energy has effectively reduced errors in atomic forces, as observed in the SPICE database.

In summary, the application of the GB-FFs model for FF parameterization represents a substantial advancement. This study presents an efficient and automated solution, improving parameter quality and saving significant time compared to conventional methods. This approach

highlights the potential of ML-based models to revolutionize the field of FF parameterization.

Of course, it is always desirable to develop newer, faster and improved model architectures. In this context, the following directions are worth considering:

- Gathering more high-quality data, as our GB-FFs model is entirely data-driven and the quality of the database significantly impacts the final performance.
- Integrating physics and chemistry into functional forms to propose the next generation of FFs.

Chapter 1

Polynomial Fit by Neural Networks

Polynomial fitting is a fundamental problem in numerical analysis and plays a critical role in various scientific and engineering applications. The conventional approaches to polynomial interpolation involve determining the coefficients of polynomials that passes through a given set of points. However, this method can be numerically unstable, especially when the points are equaly spaced and the degree of the polynomial is high.

In their recent work, Chahrour et al. compare four interpolation and three Machine Learning (ML) techniques in their recent works [64]. They find that traditional interpolation methods, like Radial Basis Function, exhibit remarkable performance within low dimensions (d = 3). In contrast, in higher dimensions (d = 5, 6, 9), Neural Networks (NNs) display greater potentials against the curse of dimensionality, offering rapid and precise predictions. In the doamine of numerical analysis, considerable researches have been devoted to the application of NNs in solving parameterized partial differential equations [65, 94] and simulating dynamic systems [30, 32]. These efforts demonstrate the potential of ML within the context of numerical simulations. However, these methods primarily focus on deriving equation solutions or system states directly from input data, whereas NNs primarily aim to learn from data samples, subsequently enabling interpolation or extrapolation.

In this chapter, we focus on using NNs to propose polynomial approximation of families of functions in a given parameter dependent family. More precisely, the goal is to predict the coefficients of polynomial interpolation in a given polynomial basis set, and not directly the value of the approximation. In contrast to conventional interpolation techniques, NNs can learn and store the general features of selected functions. Furthermore, NNs can be trained using stochastic gradient descent, which makes them scalable and efficient.

Nevertheless, the stability of NNs-based interpolation methods is still an open research question. Even minor perturbations in input data or alterations to network weights can result in significant output variations, reducing accuracy and reliability. To address this issue, we define the Lebesgue constants with perturbation (LCP) to evaluate the stability of the NNs used for polynomial interpolation. The LCP provides a quantitative measure of the quality of polynomial approximations and can be used to estimate the error of the approximation.

Our approach has several advantages over existing methods:

- Using NNs to predict the coefficients of polynomial interpolation in a given basis is a novel application of ML that has not been explored extensively.

- Our method is flexible and can be used for a wide range of interpolation problems.

- By using the Lebesgue constants, we can provide a rigorous and quantitative evaluation of the stability of NNs for polynomial interpolation.

This chapter is structured as follows: we begin by revisiting the definitions related to polynomial interpolations and introduce the LCP in Section 1.1. Subsequently, in Section 1.2.1, we assess the accuracy, additivity, and denoising capabilities of NNs interpolation for the one-dimensional case. Similar evaluations are performed for the two-dimensional case in Section 1.3. We present our conclusions in Section 1.4. Finally, there are some complementary experiments in Section 1.5, to evaluate the generalizability of this methodology.

1.1 Reminder on Classical Notations for One-Dimensional Problem

In the context of One-dimensional (1D) case, polynomial approximation is using polynomials of given degree to make the approximation of a function from the data of M point-wise. An important aspect is provided by interpolation that, from M values of a function f, proposes a polynomial approximation of degree N, with $(N+1) \leq M$:

$$I_N(f) = \sum_{i=0}^{N} \alpha_i \ell_i \tag{1.1}$$

Here are some notations that will be used in this section:

- f is the function to approximate, assumed to be continuous on its domain of definition [-1, 1]
- N is the degree of polynomials.
- M is the number of interpolation points. Usually, we consider the case $M \geq N+1$.
- $\{\ell_i | i = 0, ..., N\}$ are the basis functions to do interpolation.
- $\{x_i | i = 0, ..., M-1\}$ are the points where f is sampled.
- $\{\alpha_i | i = 0, ..., N\}$ are the coefficients of interpolation in the basis $\{\ell_i | i = 0, ..., N\}$.
- I_N is the classical polynomial interpolation of degree N when $M = N+1$.
- Λ_N are the Lebesgue Constants.
- $\Lambda_N^{\eta,p}$ are the Lebesgue Constants with Perturbation (LCP).
- $\|\cdot\|_{L^\infty}$ is the infinity norm and $\|\cdot\|_{L^2}$ is the L^2 norm.

1.1.1 Interpolation Nodes

For $M \in \mathbb{N}$, let $X = \{x_i | i = 0, 1, ..., M-1\}$ be a set of interpolation points on the real interval $\Omega = [-1, 1]$ such that $-1 \leq x_0 < x_1 < ... < x_{M-1} \leq 1$. In this chapter we consider two classical cases:

Equidistant Nodes (Case E): $x_i = -1 + \frac{2i}{M-1}$, $i = 0, 1, ..., M-1$

Chebyshev Nodes (Case C): Chebyshev points of the second kind (also called as Chebyshev extreme points, or Chebyshev–Lobatto points [95]). $x_i = \cos\theta_i$ with $\theta_i = \pi - \frac{i\pi}{M-1}$, $i = 0, 1, ..., M-1$. See Figure 1.1.

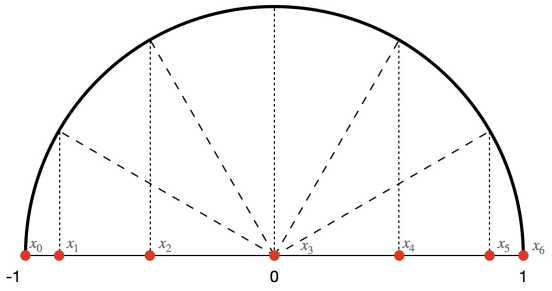

Figure 1.1: **Example of chebyshev points.** $M = 7$

1.1.2 Polynomials Functions

Before doing polynomial interpolation, we need to define the basis functions. There are two possibilities:

Lagrange Polynomials [96, 97]: When $M = N+1$, we can introduce the polynomials $\{\ell_i(x) = \prod_{k=0,k\neq i}^{N} \frac{x-x_k}{x_i-x_k} | i = 0, 1, ..., N\}$. The polynomial function ℓ_i equals 1 only at point x_i and equals 0 at all other interpolation points. These polynomials are directly decided by the values of f on interpolation nodes:

$$I_N(f) = \sum_{i=0}^{N} f(x_i)\ell_i \tag{1.2}$$

Legendre Polynomials [97, 98]: These polynomials, $\{\ell_i | i = 0, 1, ..., N\}$, satisfy that the degree of ℓ_i is i and $\int_{-1}^{1} \ell_i(x)\ell_j(x)dx = \frac{2}{2i+1}\delta_{ij}$ (where δ_{ij} denotes the Kronecker delta, equal to 1 if $i = j$ and to 0 otherwise). The polynomials $\{\ell_i | i = 0, 1, ..., N\}$ can be obtained by a Schmidt orthogonalisation process from the canonical basis $\{1, x, x^2, .., x^N\}$. In contrast to the Lagrange polynomials, Legendre polynomials are independent of interpolation nodes. Firstly we define $\ell_0(x) = 1$ and $\ell_1(x) = x$. The other polynomial functions are derived from $(i + 1)\ell_{i+1}(x) = (2i + 1)x\ell_i(x) - i\ell_{i-1}(x)$.

In practice, the number of interpolation points M is larger than the degree of polynomials: $M \geq (N + 1)$. For the Legendre polynomials, there is no restriction on interpolation points and we consider three possible cases to evaluate the influence of interpolation points on the interpolation performance: $M = N + 1, 2(N + 1), 3(N + 1)$.

When $M > N+1$, we cannot calculate the inverse of matrix to get the coefficients for Legendre polynomials. Therefore, we apply Moore-Penrose pseudo-inverse [1] [99, 100] to compute a "best fit" (least squares) solution to a system of linear equations that lacks a solution.

1.1.3 Analysis of the Stability Properties of I_N

The Lebesgue constants are defined by:

Definition 1. $\Lambda_N := \Lambda_N(x_0, x_1, ..., x_N) = \max_{-1\leq x\leq 1} \sum_{i=0}^{N} |\ell_i(x)| = \|\sum_{i=0}^{N} |\ell_i(x)|\|_{L^\infty}$

[1] https://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_inverse

It is clear that Λ_N only depends on the interpolation nodes $X = \{x_i | i = 0, 1, ..., N\}$ and it is independent of the original function f. To derive the properties of Lebesgue constant, we choose the basic Lagrange polynomials $\{\ell_i, i = 0, ..., N\}$. Supposing the polynomial approximation:

$$I_N(f)(x) = \sum_{i=0}^{N} \alpha_i \ell_i(x) = \sum_{i=0}^{N} f(x_i)\ell_i(x) \tag{1.3}$$

satisfying $I_N(f)(x_k) = f(x_k)$ for $k = 0, 1, ..., N$ where $\{x_k | k = 0, ..., N\}$ are interpolation points and $\{\alpha_i | i = 0, ..., N\}$ are the interpolation coefficients.

There exists a set of coefficients $\{\alpha_i^*\}$ achieving the smallest interpolation error in range $\Omega = [-1, 1]$:

$$\|f - \sum_{i=0}^{N} \alpha_i^* \ell_i\|_{L^\infty} = \min_{\{\alpha_i\}} \|f - \sum_{i=0}^{N} \alpha_i \ell_i\|_{L^\infty} \tag{1.4}$$

and the best polynomial approximation is noted as $I_N^*(f) = \sum_{i=0}^{N} \alpha_i^* \ell_i$

The importance of Lebesgue constant relies on the estimation of upper bound for interpolation error [101, 102]:

Proposition 1. *The approximation performance is bounded by the error of the best approximation functions:* $\|f - I_N(f)\|_{L^\infty} \leq (1 + \Lambda_N)\|f - I_N^*(f)\|_{L^\infty}$

Proof. [103]

$$\begin{aligned}
|I_N^*(f)(x) - I_N(f)(x)| &= |\sum_{i=0}^{N} I_N^*(f)(x_i)\ell_i(x) - \sum_{i=0}^{N} f(x_i)\ell_i(x)| \\
&= |\sum_{i=0}^{N} (I_N^*(f)(x_i) - f(x_i))\ell_i(x)| \\
&\leq \sum_{i=0}^{N} |\ell_i(x)| \cdot \max_{i=0,1,...,N} |I_N^*(f)(x_i) - f(x_i)| \\
&\leq \sum_{i=0}^{N} |\ell_i(x)| \cdot \|I_N^*(f) - f\|_{L^\infty}
\end{aligned}$$

Thus,

$$\|I_N^*(f) - I_N(f)\|_\infty \leq \Lambda_N \|f - I_N^*\|_{L^\infty}$$

Finally,

$$\begin{aligned}
\|f - I_N(f)\|_{L^\infty} &= \|f - I_N^*(f) + I_N^*(f) - I_N(f)\|_{L^\infty} \\
&\leq \|f - I_N^*(f)\|_{L^\infty} + \|I_N^*(f) - I_N(f)\|_{L^\infty} \\
&\leq \|f - I_N^*(f)\|_{L^\infty} + \Lambda_N \|f - I_N^*(f)\|_{L^\infty} \\
&\leq (1 + \Lambda_N)\|f - I_N^*(f)\|_{L^\infty}
\end{aligned}$$

□

There are many studies on the behavior of the Lebesgue constants corresponding to different sets of interpolation points. Equidistant points (noted as E_N) are not a good choice for polynomial interpolation and its poor behavior is explained by the Runge's phenomenon [104]. Lebesgue constants $\Lambda_N(E_N)$ grow exponentially with the asymptotic estimate [105, 106] (see Figure 1.2(Left)):

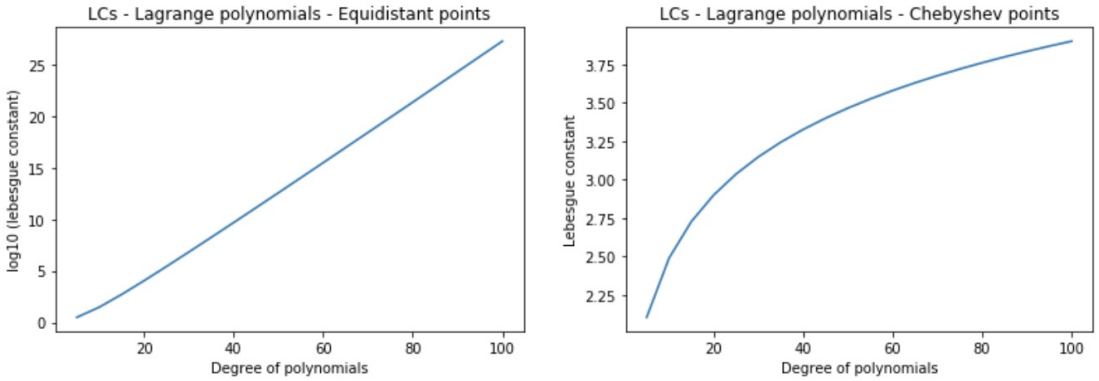

Figure 1.2: **Numerical results of Lebesgue constants on Lagrange polynomials. (Left)** Lebesgue constants grow exponentially for Equidistant points **(Right)** Lebesgue constants grow logarithmically for Chebyshev points

$$\Lambda_N(E_N) \simeq \frac{1}{e}\frac{2^{N+1}}{N(\log N + \gamma)},\ N \to +\infty \tag{1.5}$$

with $e = 2.71828...$ is the mathematical constant and γ is defined as:

$$\gamma = \lim_{N\to+\infty}(\sum_{i=1}^{N}\frac{1}{i} - \log N) \simeq 0.577 \tag{1.6}$$

The set of Chebyshev nodes is a better choice for polynomial interpolation and the Lebesgue constants $\Lambda_N(C_N)$ for polynomial interpolation grows logarithmically [107–109] (see Figure 1.2(Right)):

$$\Lambda_N(C_N) \simeq \frac{2}{\pi}\log(N),\ N \to +\infty \tag{1.7}$$

For other set of interpolation points $\{x_i\}$, see [107] for explicit formula of Lebesgue constants.

1.1.4 Lebesgue Constants with Perturbation

In this subsection, we introduce a different (though related) notion of stability constant. We start with a proposition:

Proposition 2. *The Lebesgue constant is equal to the L^∞ norm of the interpolation operator I_N:* $\Lambda_N = \max_{f\in L^\infty}\frac{\|I_N(f)\|_{L^\infty}}{\|f\|_{L^\infty}}$

Proof. [103]

$$\begin{aligned}\|I_N(f)\|_\infty &= \|\sum_{i=1}^{N} f(x_i)\ell_i\|_{L^\infty}\\ &\leq \max_{i=0,\dots,N}|f(x_i)|\|\sum_{i=1}^{N}\ell_i\|_{L^\infty}\\ &\leq \Lambda_N\|f\|_{L^\infty}\end{aligned} \tag{1.8}$$

In order to get the equality, we can construct a function f such that $\|I_N(f)\|_\infty = \Lambda_N \|f\|_{L^\infty}$:

a) Consider the function $\Phi = \sum_{i=0}^{N} |\ell_i(x)|$. Thus $\Lambda_N = \|\Phi\|_{L^\infty}$.

b) Let x^* denote a point in [-1,1] where $|\Phi(x^*)| = \|\Phi\|_{L^\infty}$.

c) Define f to take the value ± 1 at every point in $\{x_i, i = 0, ..., N\}$. f is $+1$ if $\ell_i(x^*)$ is positive and -1 if $\ell_i(x^*)$ is negative.

d) Define such a function f, in other points than the $\{x_i, i = 0, ..., N\}$ by affine interpolation so that f is continuous and defined in [-1,1] with $\|f\|_{L^\infty} = 1$.

e) Then $I_N(f)(x^*) = \sum |\ell_i(x^*)| = \Phi(x^*)$. Thus $||I_N(f)||_{L^\infty} \geq |I_N(f)(x^*)| = |\Phi(x^*)| = \Lambda_N$

Therefore, there exists a function $f \in L^\infty$ satisfying $\frac{\|I_N(f)\|_{L^\infty}}{\|f\|_{L^\infty}} = \Lambda_N$. □

An important issue in numerical analysis is the stability of the objects that are introduced. Here we wonder what is the effect, on the interpolation polynomial, of the imprecision on the values of $f(x_i), i = 0, .., N$. We thus consider the case where the function f is contaminated by noise ε (in this section, we use random noise) and we measure the effect of this noise in L^p-norm ($p \in \{2, \infty\}$). From the additivity of the interpolation operator I_N

$$\begin{aligned} \|I_N(f+\varepsilon) - I_N(f)\|_{L^p} &= \|I_N(\varepsilon) + I_N(f) - I_N(f)\|_{L^p} \\ &= \|I_N(\varepsilon)\|_{L^p} \\ &= \|\sum_{i=0}^{N} \varepsilon(x_i)\ell_i(x)\|_{L^p} \\ &\leq \|\varepsilon\|_{L^\infty} \|\sum_{i=0}^{N} |\ell_i|\|_{L^p} \end{aligned} \tag{1.9}$$

In what follows, we shall introduce different interpolation methods, either for $M = N + 1$ or more generally for $M \geq N+1$, that may not even be linear. Let P_N denote such an interpolator, we can define the Lebesgue Constants with Perturbation (LCP) $\Lambda_N^{\eta,p}$ as:

$$\Lambda_N^{\eta,p} := \max_{f,\varepsilon} \frac{\|P_N(f+\varepsilon) - P_N(f)\|_{L^p}}{\|\varepsilon\|_{L^\infty}} \tag{1.10}$$

with $p \in \{2, \infty\}, f \in L^p$, $P_N = I_N$, $\frac{\|\varepsilon\|_{L^\infty}}{\|f\|_{L^\infty}} \leq \eta$ and the interval is still $[-1, 1]$. η is the level of noise.

Obviously, when $p = \infty$, Equation (1.9) gives $\Lambda_N^{\eta,\infty} \leq \Lambda_N$. Additionally, due to the definition of L^∞ norm, we can get $\Lambda_N^{\eta,2} \leq \Lambda_N^{\eta,\infty} \leq \Lambda_N$

We are going to use $\Lambda_N^{\eta,p}$ to evaluate the stability of polynomials interpolation by NNs that is introduced in the next sections. In this chapter, we have considered three levels of noise: $\eta = 0, 0.1\%, 10\%$. A smaller value of $\Lambda_N^{\eta,p}$ indicates the stronger noise resistance of the model.

1.2 Experiments for One-Dimensional Cases

1.2.1 One Dimensional Function

We choose the parameter dependent function:

$$f(x;\mu=[\theta_1,\theta_2,\theta_3]) = (\theta_1 x + \sin(\theta_2 x))e^{-\theta_3 x} \tag{1.11}$$

with $\mu \in [-1,1]\times[-3,3]\times[0.5,5]$ and $x \in [-1,1]$.

Considering the range of function values is too big, the functions are scaled to satisfy $\max_{i\in[0,M-1]}|f(x_i)| = 1$.

1.2.2 Neural Networks Architecture

We use NNs to replace the classical polynomial interpolation. Note G_N as the NNs interpolation operator. Similar to the polynomial interpolation operator I_N, G_N takes the function values (with or without noise) on interpolation points as inputs and outputs the corresponding coefficients $\{\alpha_i^{NNs}, i = 0,1,...,N\}$ (see Figure 1.3), from which we can construct an approximation:

$$G_N(f) = \sum_{i=0}^{N} \alpha_i^{NNs}\ell_i \tag{1.12}$$

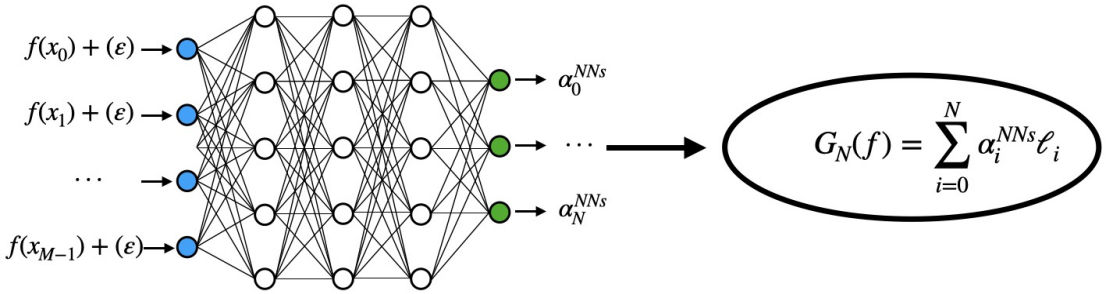

Figure 1.3: **Framework of 1D NNs interpolation G_N.** M is the number of interpolation points. N is the degree of polynomials. Input data is the function values on interpolation points with noise. Neural networks will give the coefficients $\{\alpha_i^{NNs}\}$. As the basis functions $\{\ell_i\}$ are pre-defined (Lagrange polynomials or Legendre polynomials), we can finally construct an approximation.

As shown in Figure 1.3, the number of inputs ($M \leq 303$) and outputs($N+1 \leq 101$) are fixed, we can simply apply the feed-forward NNs to construct interpolation operator G_N. G_N contain 5 layers. Except the input layer (blue nodes in Figure 1.3) and the output layer (green nodes in Figure 1.3), there are still three hidden layers (white nodes in Figure 1.3) with each hidden layer containing 128 neurons.

The model parameters are initialized with Xavier initialization [110], an uniform distribution for *tanh* activation function. Between two connected layers, there exists an activation function to ensure the non-linearity. The common activation functions, such as REctified Linear Unit (*Relu*) and hyperbolic tangent (*tanh*), work well at most cases and our models use *tanh* function:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.13}$$

Thus the whole process of G_N is:

$$G_N(f) = W_5 \cdot tanh(W_4 \cdot tanh(W_3 \cdot tanh(W_2 \cdot tanh(W_1 \cdot f(X))))) \tag{1.14}$$

with $W_1 \in \mathbb{R}^{128\times M}, W_2, W_3, W_4 \in \mathbb{R}^{128\times 128}, W_5 \in \mathbb{R}^{(N+1)\times 128}$.

Number of inputs	$M \leq 303$	Initial learning rate	1E-03	Training data	160,000
Number of outputs	$N \leq 100$	Smallest learning rate	5E-05	Validation data	8,000
Dimension of model	128	Gamma	0.999	Test data	8,000
Number of layers	5	Maximum epochs	600	Batch size	10,000

Table 1.1: Hyper-parameters in neural networks and model information

To generate the training dataset for the NNs, we employ a set of functions as described in Equation (1.11). These functions depend on three parameters: θ_1, θ_2, and θ_3. We sample θ_1 from the range [-1, 1] and θ_2 from the range [-3, 3] at 40 uniformly spaced points. Additionally, θ_3 is sampled at 100 uniformly spaced points within the range of 0.5 to 5. In total, we generate 160,000 functions for each unique combination of these parameters, which constitutes our training dataset.

For the construction of the validation and test datasets, we adopt a similar approach. However, in contrast to the training dataset, we sample θ_1 and θ_2 at 10 uniformly spaced points within the ranges of [-1, 1] and [-3, 3], respectively, plus additional 10 points randomly selected from these intervals. For θ_3, we sample 10 points uniformly from the range 0.5 to 5, along with 10 additional points chosen at random within this interval. Each of the validation and test datasets comprises a total of 8,000 functions.

Since the output of the NNs are the coefficients, this belongs to the regression task for which we will utilize the Mean Square Error (MSE) loss:

$$MSELoss(x, y) = (x - y)^2 \tag{1.15}$$

where x is the predicted result and y is the target value.

The loss functions for one given function f are defined as the difference of interpolation values over the range Ω:

$$Loss = \sum_{k=0}^{1000} MSELoss(\sum_{i=0}^{N} \alpha_i^{NNs}\ell_i(x_k), f(x_k)) = \sum_{k=0}^{1000}((\sum_{i=0}^{N} \alpha_i^{NNs}\ell_i(x_k) - f(x_k))^2 \tag{1.16}$$

where $\{\alpha_i^{NNs}|i = 0, 1, ..., N\}$ are the coefficients predicted by NNs, $\{\alpha_i|i = 0, 1, ..., N\}$ are the coefficients obtained by interpolation, $\{\ell_i|i = 0, 1, ..., N\}$ are the polynomials and $\{x_k|k = 0, 1, ..., 1000\}$ are the equidistant sampled points in range Ω ($\forall \in [0, 999], x_{k+1} - x_k = 0.002$). It should be noted that the loss is defined on the error on 1001 sampled points instead of the interpolation points because we expect the NNs operator G_N to learn more information. We select the number 1001 is for balancing the computational efficiency and accuracy.

Above loss function is actually the L^2 norm of approximation error. We can apply Trapezoidal rule, which is more accurate, to do integration for L^2 norm and the loss function becomes:

$$\begin{aligned} Loss = &\sum_{k=1}^{999}((\sum_{i=0}^{N} \alpha_i^{NNs}\ell_i(x_k) - f(x_k))^2 \\ &+ \frac{((\sum_{i=0}^{N} \alpha_i^{NNs}\ell_i(x_0) - f(x_0))^2 + ((\sum_{i=0}^{N} \alpha_i^{NNs}\ell_i(x_{1000}) - f(x_{1000}))^2}{2} \end{aligned} \tag{1.17}$$

Indeed, our experiments show that the inclusion or exclusion of Trapezoidal approximation has a minimal impact on the final performance.

For 1D experiment, we have trained 420 models in total. In the case of Legendre polynomials, we investigate 18 different combinations, varying the number of interpolation points ($M = N+1$, $M = 2(N+1)$ or $M = 3(N+1)$), interpolation point types (Equidistant or Chebyshev), and input data noise levels ($\eta = 0, 0.1\%, 10\%$). For Lagrange polynomials, we are constrained to use $M = N + 1$, resulting in six possible combinations (three levels of noise for Equidistant points and also three for Chebyshev points).

However, we find that the combination of Equidistant points and Lagrange polynomials requires higher precision of float number because the polynomials sometimes contain extreme large value (due to the Runge phenomenon) around the endpoints. According to our test, "float 32", which is a 32 bit number, can only provide enough precision for $N \leq 40$ while "float64" can provide enough precision for $N \leq 70$. Therefore, we get rid of the three combinations related to Equidistant points and Lagrange polynomials.

In summary, there are 21 combinations. For each combination, we have trained 20 models by varying the polynomial dimension from 5 to 100 with a step size of 5. Therefore, the total number of models trained is 420.

Models are trained on 1 NVIDIA V100 GPU of 16GB. Each model took about twenty minutes to train. As our models are very small (about 230 KB), there is no warmup. Initial learning rate for all parameters is 0.001, that decays by gamma (see Table 1.1) every 50 steps. The decay will be stopped after learning rate (inTable 1.1) is less than smallest learning rate. Additionally, we used the Adam optimiser [111] with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is set to 10,000 to accelerate the training stage. The training stage lasts at most 600 epochs and it will be early stopped if the loss for validation dataset does not decrease in consecutive 15 epochs.

1.2.3 Accuracy of Neural Networks

In this chapter, our primary focus lies in employing NNs to predict the coefficients for polynomial fitting. We begin by proving that the NNs operator, denoted as G_N, can give a fair enough approximation. Our experiments shows that the accuracies of the 21 combinations are comparable. Therefore, here we exclusively present the results for the case $M = N + 1$ trained by noise-free ($\eta = 0$). Figure 1.4 presents the approximation results on noise-free test data ($\eta = 0$).

Figure 1.4 shows the error norm for approximation. The horizontal axis represents the degree of polynomials, and the vertical axis represents $\log\|P_N(f) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or NNs interpolation G_N (full line). The results for Equidistant points and Legendre polynomials are in blue. The results for Chebyshev points and Legendre polynomials are in green. The results for Chebyshev points and Lagrange polynomials are in red.

Firstly, the case of classical interpolation I_N for Equidistant points and Legendre polynomials (blue dot line) is the worst, as its accuracy decreases with the number of degrees of freedom (N) after $N \geq 25$ due to the Runge phenomenon [104]. Using the NNs for interpolating G_N can avoid this problem because our loss function 1.17 is defined for 1001 equidistant sampled points rather than the values on interpolation points M, the abnormal values around end-points are avoided.

The use of Chebyshev points is effective in avoiding the Runge phenomenon, and as N increases, the approximation improves, but the error norm stabilizes due to the limitations of floating-point precision. Unexpectedly, for N greater than 10, the accuracy of G_N is significantly lower than that of I_N. We analyze this discrepancy as being caused by the fact that we are predicting the coefficients rather than directly predicting the function values. A small variation in coefficients bring effective change in the final performance. Our results show that NNs are not

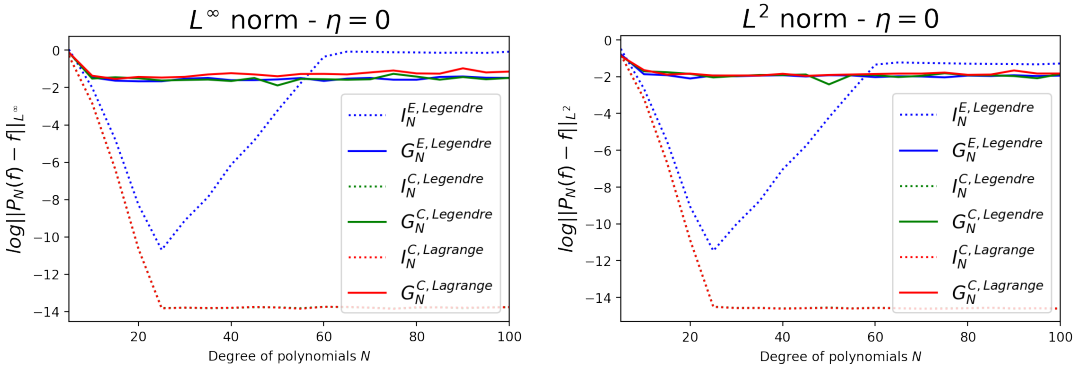

Figure 1.4: **Accuracy of one-dimensional approximation interpolation** ($M = N + 1, \eta = 0$)**.** $\log\|P_N(f) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or NNs interpolation G_N (full line). **(Left)** L^∞ norm . **(Right)** L^2 norm.

sensitive to the type of interpolation points or the type of polynomials.

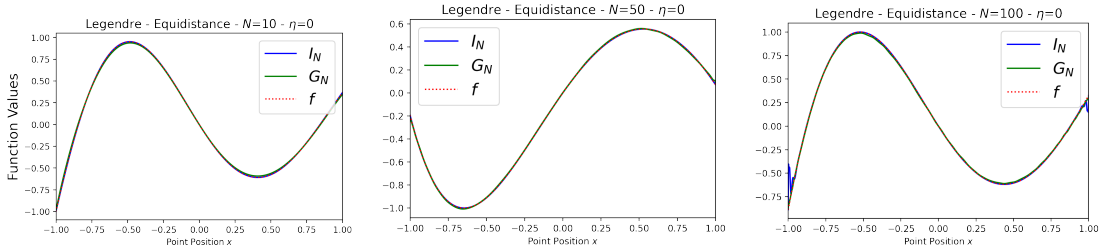

Figure 1.5: **Examples of approximation accuracy. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $M = 100$.

Even G_N are not as good as I_N, they still give good enough approximation if we are interested with accuracies that ranges between 10^{-1} and 10^{-2}. In Figure 1.5, we show three examples of the case: Legendre polynomials, Equidistant points and $M = N + 1$, so you can observe the Runge phenomenon when $N = 100$. The chosen functions satisfy:

$$f = \arg\max_{f \in L^\infty} \|G_N(f) - f)\|_{L^\infty} \tag{1.18}$$

I_N is in blue. G_N is in green and the original function values are the red dot lines. When $N = 100$, the Runge phenomenon is obvious (Figure 1.5 (Right)).

From the approximation point of view, it is disappointing to find that ML is not as powerful as we imagine to do parameterization tasks. Even if NNs provide fair enough approximation, they are still far beyond the classical interpolation. The approximation performance may be optimized if we apply other special ML technologies. However, this part aims to prove the feasibility of processing parameterization tasks by NNs. Some interesting features are however coming in the next analysis.

1.2.4 Additivity of Neural Networks

NNs consist of non-linear activation functions and linear combinations. Thus the NNs are naturally non-linear operator. However, when it is applied to do the polynomial fitting, it is expected to be close to a linear operator. That is why we checkout the additivity of G_N.

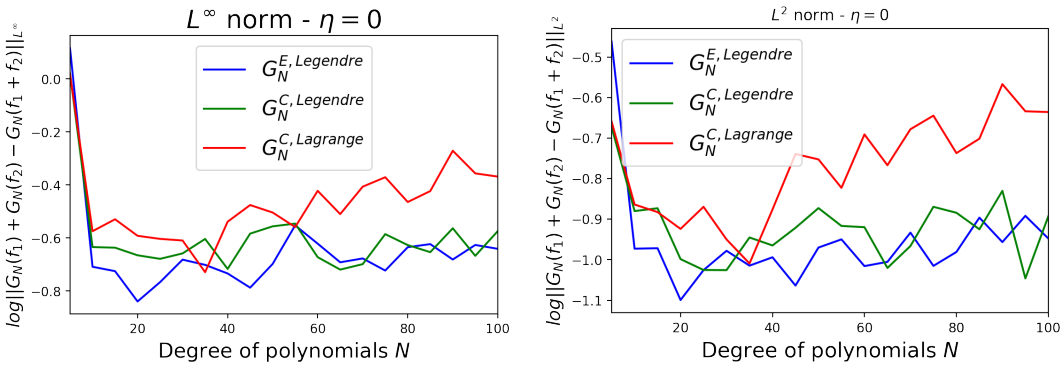

Figure 1.6: **Additivity of G_N.** $\log\|G_N(f_1) + G_N(f_2) - G_N(f_1 + f_2)\|_{L^p}$ where $p \in \{2, \infty\}$. **(Left)** L^∞ norm. **(Right)** L^2 norm.

Similarly to the last subsection, the models are trained by noise-free data and only the cases $M = N + 1$ are considered. The test data also has no noise ($\eta = 0$). The functions f_1 and f_2 are both from the test dataset. Figure 1.6 shows the error for additivity. The horizontal axis represents the degree of the polynomials, and the vertical axis represents the $\log\|G_N(f_1) + G_N(f_2) - G_N(f_1 + f_2)\|_{L^p}$ where $p \in \{2, \infty\}$. The results for Equidistant points and Legendre polynomials are in blue. The results for Chebyshev points and Legendre polynomials are in green. The results for Chebyshev points and Lagrange polynomials are in red.

From Figure 1.6, when $n = 5$, as the approximation is bad, the additivity is never preserved. When N increases, the additivity is obtained at the level of accuracy that G_N is able to achieve, at least in the case of Legendre basis set. The combination for Lagrange polynomials and Chebyshev points (red lines) has the worst performance in terms of additivity.

In Figure 1.7, we have showed three example under the condition of Legendre polynomials and Chebyshev points for $N = 10, 50, 100$. The functions f_1, f_2 from test dataset are chosen with maximum L_∞ error:

$$f_1, f_2 = \underset{f_1, f_2 \in L^\infty}{\arg\max} \|G_N(f_1) + G_N(f_2) - G_N(f_1 + f_2)\|_{L^\infty} \tag{1.19}$$

The blue lines represent the sum of two original functions, the orange lines represents the new functions obtained by fitting the sum of these two functions, and the green lines are the sum of the two functions fitted separately. The closer the orange and green lines are, the more G_N preserves additivity.

1.2.5 Ability of Denoising

We are also curious about models' ability to filter noise during interpolation. Here we only present the results for combination Chebyshev points and Legendre polynomials for $M = N + 1$. In fact, for other combinations, the results are similar.

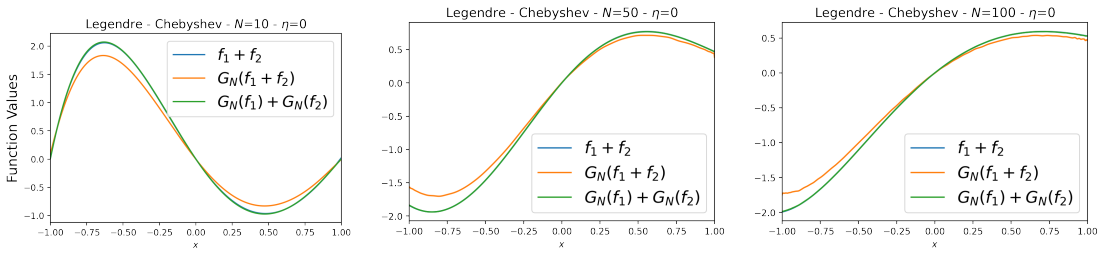

Figure 1.7: **Example of additivity for G_N. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $N = 100$.

The introduction of noise involves adding a random noise value to the original function value. The test data contains noise ($\eta = 10\%$). The norm $\log\|P_N(f+\varepsilon) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be I_N (black dot lines) or G_N (full lines) are shown in Figure 1.8.

G_N trained by data with noise $\eta = 0.1\%$ (green lines) and without noise(blue lines) have no significant difference. This is because the noise in level $\eta = 0.1\%$ is negligible. The model trained by data with noise $\eta = 10\%$ shows the strongest ability to filter noise. Therefore, taking denoising task into account during training stage enhances models' ability to filter noise. And G_N always performs better than I_N.

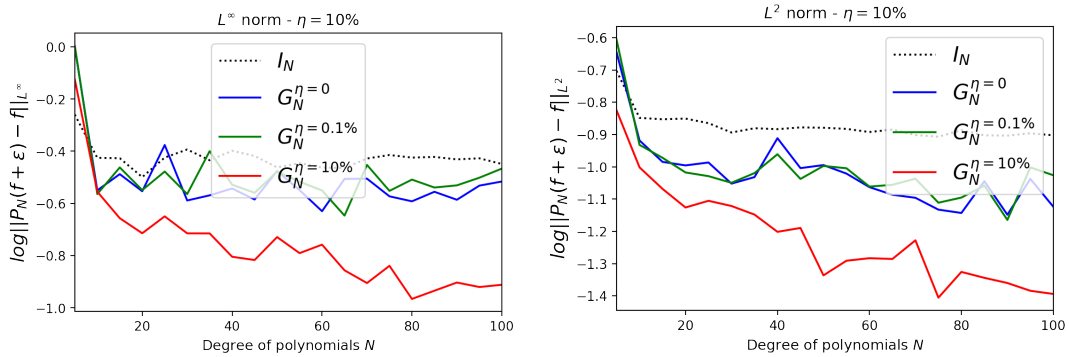

Figure 1.8: **Denoising of G_N.** $\log\|P_N(f+\varepsilon) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or neural networks interpolation G_N (full line). **(Left)** L^∞ norm. **(Right)** L^2 norm.

In Figure 1.9, there are three examples for the case: Chebyshev points, Legendre polynomials, $M = N + 1$ and $N = 100$ in Figure 1.9. And the functions satisfy:

$$f = \arg\max_{f \in L^\infty} \|G_N^\eta(f+\varepsilon) - f)\|_{L^\infty} \tag{1.20}$$

I_N tend to retain the noise while G_N is able to filter the fluctuations.

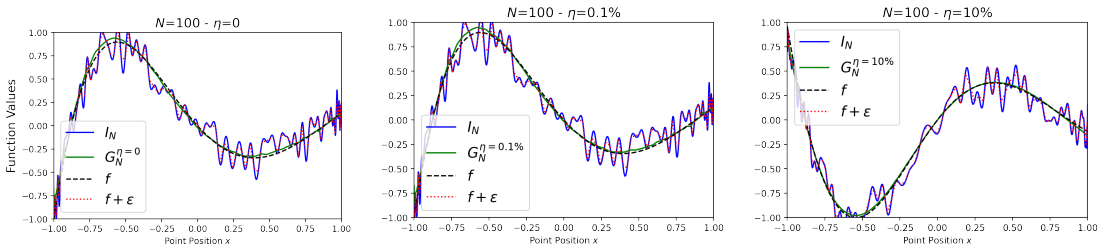

Figure 1.9: **Example of denoising. (Left)** $G_N^{\eta=0}$. **(Middle)** $G_N^{\eta=0.1\%}$. **(Right)** $G_N^{\eta=10\%}$.

1.2.6 Evaluation of $\Lambda_N^{\eta,p}$

We now move on to the analysis of $\Lambda_N^{\eta,p}$. First, we use the $\Lambda_N^{\eta,\infty}$ defined by Equation (1.10). We start by considering Legendre polynomials, Equidistant points. The results are shown in Figure 1.10. The x-axis represents the polynomial degree N, while the y-axis represents $\Lambda_N^{\eta,p}$. The blue, orange, and green lines represent the results of models trained on data with noise levels of $\eta = 0, 0.1\%, 10\%$ respectively, while the red lines represent the results obtained by polynomial interpolation I_N (for I_N, it is in $\log \Lambda_N^{\eta,p}$).

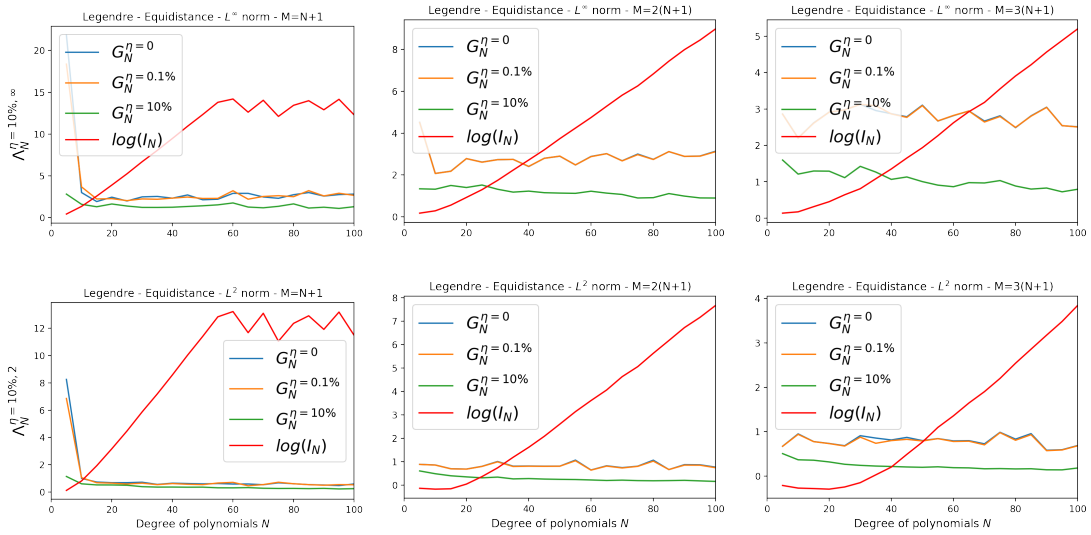

Figure 1.10: $\Lambda_N^{\eta,p}$ **for Legendre polynomials and Equidistant points. (Left:)** $M = N+1$. **Middle** $M = 2(N+1)$. **(Right)**$M = 3(N+1)$. **(Top)**$\Lambda_N^{\eta,\infty}$. **(Bottom)**$\Lambda_N^{\eta,2}$

The results in Figure 1.10 confirm that Equidistant point is a poor choice for interpolation, as $\Lambda_N^{\eta,\infty}$ increase exponentially with the polynomial degree N. The latter part of red line for $M = N+1$ stops increasing due to the limit of float number's precision. Recalling the Lebesgue constants in Figure 1.10(Left), we find that Λ_N and $\Lambda_N^{\eta,\infty}$ have similar behaviors in this case. Additionally, as the red lines are almost always above the other three lines, it suggests that, compared to direct interpolation, using G_N to predict the interpolation coefficients improves the stability.

When using Chebyshev nodes, $\Lambda_N^{\eta,\infty}$ remain stable and do not vary with the polynomial degree N. The impact of training data on the stability of the models is also significant. As the proportion of noise in the training data increases, the stability of different models to the same set of noisy data ($\eta = 10\%$) is also strengthened, with the green lines ($G_N^{\eta=10\%}$) in Figure 1.11 showing the smallest $\Lambda_N^{\eta,\infty}$. The orange lines represent the models trained on data with only very little noise ($\eta = 0.1\%$), their performance is similar to that of noise-free model.

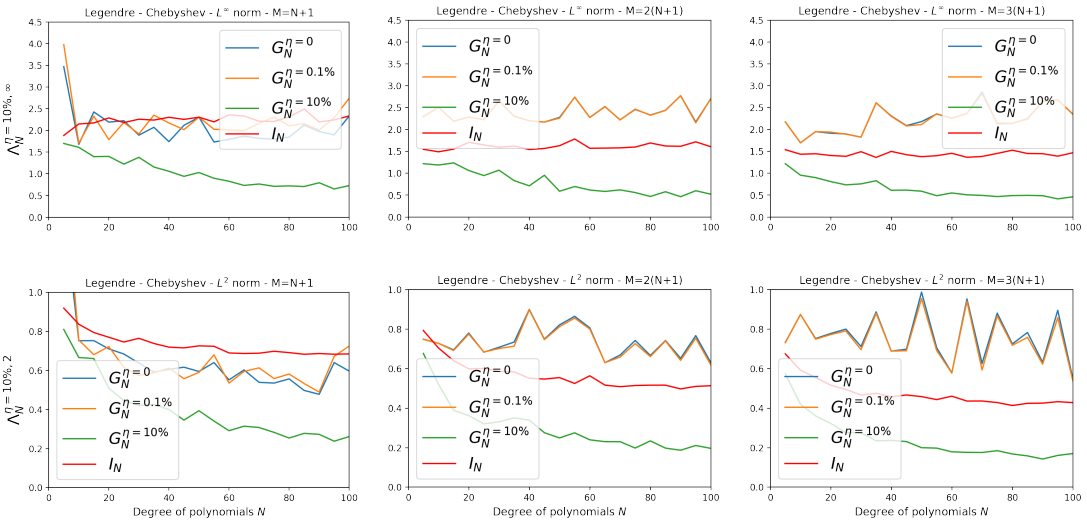

Figure 1.11: $\Lambda_N^{\eta,p}$ **for Legendre polynomials and Chebyshev points. (Left)** $M = N + 1$. **Middle** $M = 2(N + 1)$. **(Right)**$M = 3(N + 1)$. **(Top)**$\Lambda_N^{\eta,\infty}$. **(Bottom)**$\Lambda_N^{\eta,2}$

In Figure 1.11(Top three sub-figures), the $\Lambda_N^{\eta,\infty}$ from G_N and I_N have fluctuations in a range. Increasing the number of interpolation points M without increasing the polynomial dimension N has improvement on I_N. The performance of polynomial interpolation (red lines) is better than $G_N^{\eta=0}$ (blue lines, model trained by data without noise) and $G_N^{\eta=0.1\%}$ (orange lines, model trained by data with noise $\eta = 0.1\%$) for $M \geq 2(N + 1)$ but it is not as good as $G_N^{\eta=10\%}$ (green lines, model trained by data with noise $\eta = 10\%$). Therefore, it proves that adding noise to training data increases the stability of interpolation in terms of noise. This conclusion corresponds to the idea in Denoising AutoEncoder [112].

Figure 1.11(Bottom three sub-figures) have presented the results for $\Lambda_N^{\eta,2}$, which focus on the anti-noise stability over all range $\Omega = [-1, 1]$. We notice that both G_N and I_N shows higher stability to noise when N increases.

Still considering the Chebyshev points, the results for Lagrange polynomials (Figure 1.12) are similar to the results for case $M = N + 1$ in Legendre polynomials (Figure 1.11(Left)). It means the interpolation performance of G_N is theoretically independent of polynomials.

Here are the conclusions for 1D case:

- NNs can be used to predict interpolant coefficients and construct approximate functions. But the accuracy is very limited.
- All combinations that contain Chebyshev points have good performance for both NNs interpolation G_N and polynomial interpolation I_N. For I_N, Equidistant points is a bad choice, in terms of stability to noise, that should be avoided.

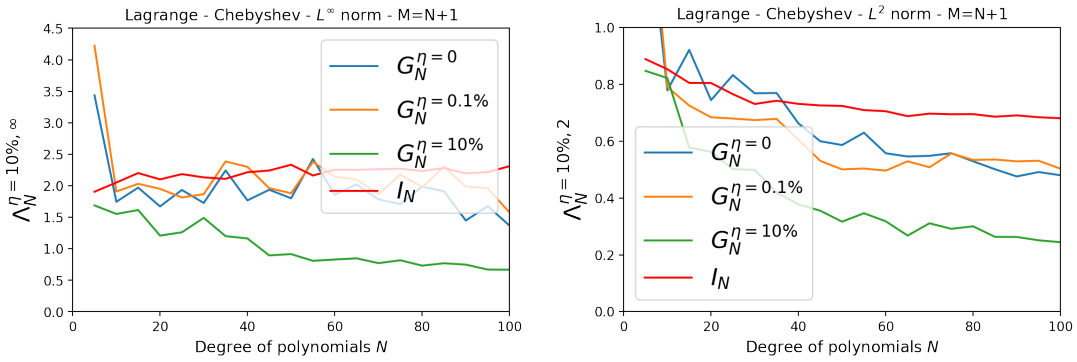

Figure 1.12: $\Lambda_N^{\eta,p}$ **for Lagrange polynomials and Chebyshev points. (Left)** $\Lambda_N^{\eta,\infty}$. **(Right)** $\Lambda_N^{\eta,2}$.

- For both G_N and I_N, in various combinations, Lebesgue constant with perturbation ($\Lambda_N^{\eta,p}$) are closely related to polynomial degree N.

- The stability of the models trained on data with noise $G_N^{\eta=10\%}$ (green lines) are always higher than polynomial interpolation I_N(red lines). If the noise in the training data is small ($\eta = 0.1\%$, orange lines) or even non-existent ($\eta = 0$, blue lines), the models' performance is usually comparable to polynomial interpolation. This suggests that the limitation of NNs interpolation: if noise filtering is not involved in the training task, the final stability against noise will be significantly reduced.

1.3 Experiments for Two Dimensional Problems

Following the structure in experiments of 1D cases, we introduce the functions to test, explain the details of NNs and compare our models with polynomial interpolation through the evaluation of LCP ($\Lambda_N^{\eta,p}$).

1.3.1 Two dimensional Functions

The two dimensional parameter dependent functions are defined as:

$$f(x_1, x_2; \mu = [\theta_1, \theta_2, \theta_3, \theta_4]) = \theta_1 e^{-((x_1-\theta_2)^2+(x_2-\theta_2)^2)} + e^{-((x_1-\theta_3)^2+(x_2-\theta_4)^2)} \tag{1.21}$$

with $\mu \in [-2, 2] \times [-1, 1] \times [-1, 1] \times [-1, 1], (x_1, x_2) \in \Omega$ and $\Omega = [-1, 1] \times [-1, 1]$.

The functions are also scaled to satisfy $\max_{(x_1,x_2)\in X} |f(x_1, x_2)| = 1$ with X is the ensemble of interpolation points. And the 2D polynomial approximation function in degree N is written as:

$$I_N(x_1, x_2) = \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha_{ij} \ell_i(x_1) \ell_j(x_2) \tag{1.22}$$

where α_{ij} are the interpolation coefficients, ℓ_i and ℓ_j are the Lagrange polynomials or Legendre polynomials defined in Section 1.1.2. Therefore, for the 2D polynomial interpolation function I_N has $(N+1)^2$ coefficients.

For Lagrange polynomials , we only consider the case that interpolation points $M = (N+1)^2$. For Legendre polynomials , we consider the cases that interpolation points M more than number of polynomial coefficients. More precisely, we consider the cases: $M = (N+1)^2, 4(N+1)^2, 9(N+1)^2$.

1.3.2 Architecture of Convolutional Neural Networks

Due to the number of coefficients increasing with the square of the dimension N, using fully connected NNs would requires training an impractically large number of parameters. Therefore, we need a more flexible NN. As both Equidistant and Chebyshev points yield regularly spaced interpolation points, we use convolutional neural networks (CNNs) [67–69] to handle this structured data.

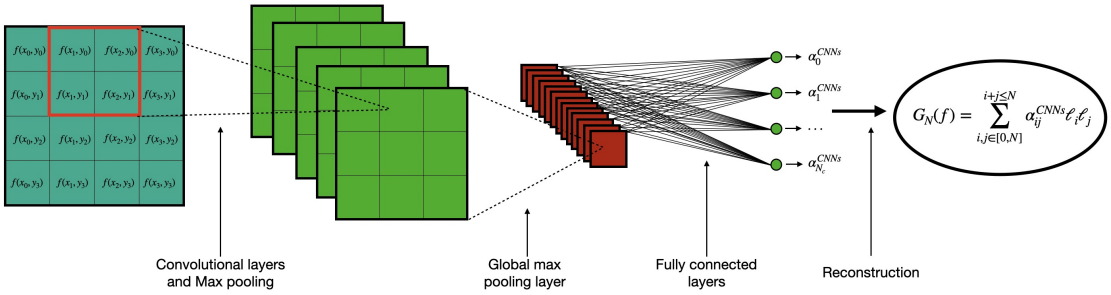

Figure 1.13: **Framework of 2D CNNs interpolation G_N.** To get the interpolation coefficients, input data will go through the convolutional layers, the Global max pooling layer and the fully connected layers.

As shown in Figure 1.13, the input data is the function values (with or without noise) on each interpolation points. After passing several convolutional layers and Max Pooling layers (MP), we apply the Global Max Pooling (GMP) [70, 71] to transform the data into a fixed-length vector. This vector can then be connected to a fully connected NNs to predict the interpolation coefficients.

MP is a down-sampling operation commonly used in CNNs. It involves dividing the input feature map into non-overlapping regions and selecting the maximum value from each region. This process reduces the spatial dimensions of the feature map, helping to retain important features while reducing computational complexity. GMP is a variation of MP where the entire feature map is thought as a single region, and the maximum value across the entire feature map is selected. This operation produces a single value for each channel, summarizing the most important information in the entire features. GMP is often used before fully connected layers.

Our CNNs architecture consists of five convolutional layers (in kernel 3×3) with `tanh` activation functions followed by max pooling (in size 2×2) layers and a Global max pooling layer. All trainable parameters are initialized by Xavier initialization [110]. The whole process of G_N is:

$$\alpha^{CNNs} = W_1 \cdot \sigma(\texttt{GMP}(\texttt{C}_5(\texttt{MP}(\sigma(\texttt{C}_4(\sigma(\texttt{C}_3(\texttt{MP}(\sigma(\texttt{C}_2(\sigma(\texttt{C}_1(f(X))))))))))))) \tag{1.23}$$

where $\texttt{C}_1, \texttt{C}_2, \texttt{C}_3, \texttt{C}_4, \texttt{C}_5$ are the Convolutional layers, $\sigma(\cdot)$ is the `tanh` activation function, `MP`

is the max pooling layer, `GMP` is the Global max pooling layer and $W_1 \in \mathbb{R}^{N_t \times (N+1)^2}$ with $N_t = \max(\frac{N^2}{2}, 128)$. Specifically, if we note the parameters in convolutional layer as $W_C \in \mathbb{R}^{C_{in} \times C_{out} \times m \times n}$, with C_{in} is the number of input channels, C_{out} is the number of output channels and $m \times n$ is the size of kernel, the parameters in five convolutional layers should be: $W_{C_1} \in \mathbb{R}^{1\times4\times3\times3}, W_{C_2} \in \mathbb{R}^{4\times8\times3\times3}, W_{C_3} \in \mathbb{R}^{8\times16\times3\times3}, W_{C_4} \in \mathbb{R}^{16\times64\times3\times3}, W_{C_5} \in \mathbb{R}^{64\times N_t\times3\times3}$.

Number of inputs	$M \leq 15,129$	Initial learning rate	5E-03	Training data	28,561
Number of outputs	$(N+1)^2 \leq 1,681$	Smallest learning rate	1E-04	Validation data	4,096
		Gamma	0.999	Test data	4,096
		Maximum epochs	600	Batch size	512

Table 1.2: Hyper-parameters in CNNs and model information

To generate the training data, we use the functions presented in Equation (1.21) with four parameters θ_1, θ_2, θ_3, and θ_4. θ_1 is uniformly sampled at 13 points in $[-2, 2]$ while the others are all uniformly sampled at 13 points in $[-1, 1]$. In total, 28,561 functions are generated for each combination of parameters for training purposes.

Similarly, we need to construct validation and test datasets. $\theta_1/\theta_2, \theta_3, \theta_4$ are uniformly sampled at 4 points in $[-2, 2]/[-1, 1]$, plus 4 randomly sampled points in $[-2, 2]/[-1, 1]$. In total, there are 4,096 functions in each of the validation and test datasets. Please check Table 1.2 for more training details.

Similarly to the 1D cases, we define the following loss function:

$$loss = \sum_{m=0}^{200}\sum_{n=0}^{200}((\sum_{i=0}^{N}\sum_{j=0}^{N}\alpha_{ij}^{CNNs}\ell_i(x_m)\ell_j(x_n) - f(x_m, x_n))^2 \tag{1.24}$$

where $\{\alpha_{ij}^{CNNs}|i, j = 0, 1, ..., N\}$ are the coefficients predicted by CNNs, $\{\ell_i|i = 0, 1, ..., N\}$ are the polynomials and $\{(x_m, x_n)|m, n = 0, ..., 200\}$ are the uniform sampled points in range $\Omega = [-1, 1] \times [-1, 1]$.

For 2D experiments, we have only trained 168 models in total. As the same to 1D cases, there are 21 combinations. For each combination, we trained 8 models by varying the polynomial dimension from 5 to 40 with a step size of 5. We give up the models with $N > 40$ due to the limitation of computational resource.

1.3.3 Accuracy of Convolutional Neural Networks Interpolation

We follow the idea in 1D cases, proving the accuracy of CNNs before other experiments. The accuracy results are almost the same as 1D cases. CNNs cannot give perfect approximation coefficients and they are still far behind the classical interpolation. But the following example shows the approximation in most area is is rather acceptable if we look for accuracies about 10^{-1} or 10^{-2}.

Here we only present an example of the Legendre polynomial with Chebyshev points and $M = (N + 1)^2$ in Figure 1.15. The models G_N are trained by data without noise ($\eta = 0$) and polynomial dimensions $N = 40$. The error norm $\log\|P_N(f) - f\|$ is only evaluated on noise-free test data ($\eta = 0$) and results are shown in Figure 1.15. The function f in example holds the largest approximation error (see Equation 1.18). In Figure 1.15(Right), there largest error exists in the four corners.

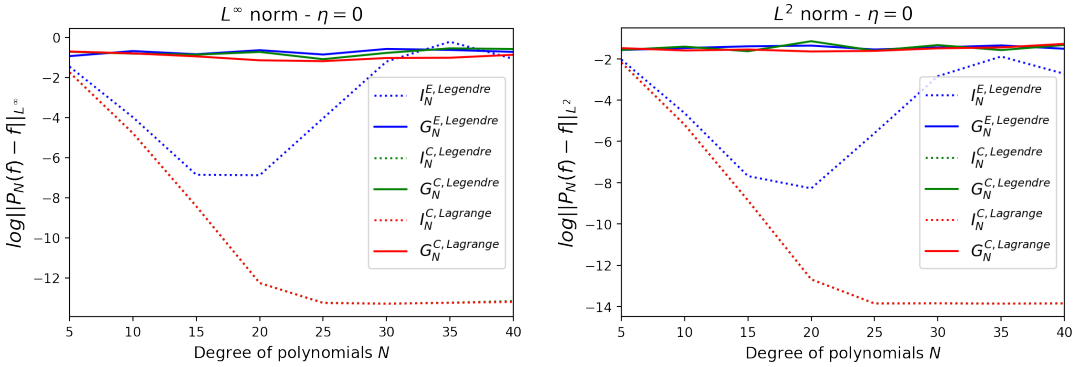

Figure 1.14: **Accuracy of two-dimensional approximation interpolation.** $log\|P_N(f) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or CNNs interpolation G_N (full line). **(Left)** L^∞ norm . **(Right)** L^2 norm.

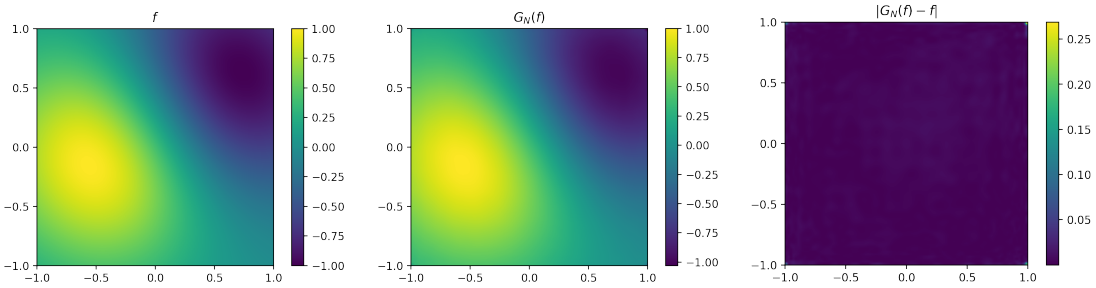

Figure 1.15: **Examples of approximation accuracy for two-dimensional case. (Left)** function values f. **(Middle)** approximation values $G_N(f)$. **(Right)** difference $|G_N(f) - f|$.

1.3.4 Ability of Denoising

Still considering the case $M = (N+1)^2$, Legendre polynomials and Chebyshev points. We use the test data with noise $\eta = 10\%$ to see G_N's ability to filter noise. Figure 1.16 shows that, with the increase of N, G_N gradually lose their denoising ability. But the coefficients provided by NNs still outperform the classical interpolation in terms of denoising.

Here is an example for Chebyshev points and Legendre polynomials in Figure 1.17. The largest error appears at left-bottom corner.

1.3.5 Evaluation of $\Lambda_N^{\eta,p}$

Next we are gonna present the results of $\Lambda_N^{\eta,\infty}$ for all models. Please notice that in Figure 1.18 and 1.19, $\Lambda_N^{\eta,\infty}$ for all models (I_N and G_N) are in *log*.

We first observe the results of Legendre polynomials in Figure 1.18 and 1.19 . Different from the conclusions in 1D condition, LCP for G_N increase with the dimension of degrees N. This is because we only employ a naive CNNs to predict polynomial coefficients. As the number of inputs (values on interpolation points) and outputs (coefficients α^{CNNs}) increases, our model

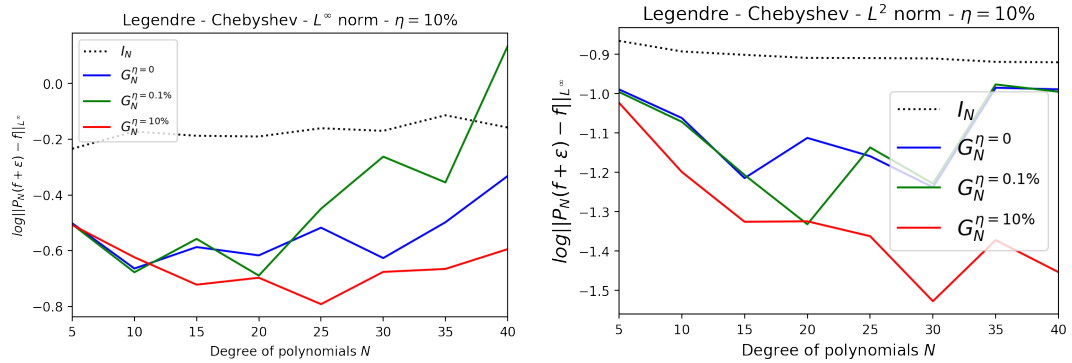

Figure 1.16: **Ability to filter noise of for two dimensional functions.** $\log\|P_N(f+\varepsilon)-f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or NNs interpolation G_N (full line). **(Left)** L^∞ norm. **(Right)** L^2 norm.

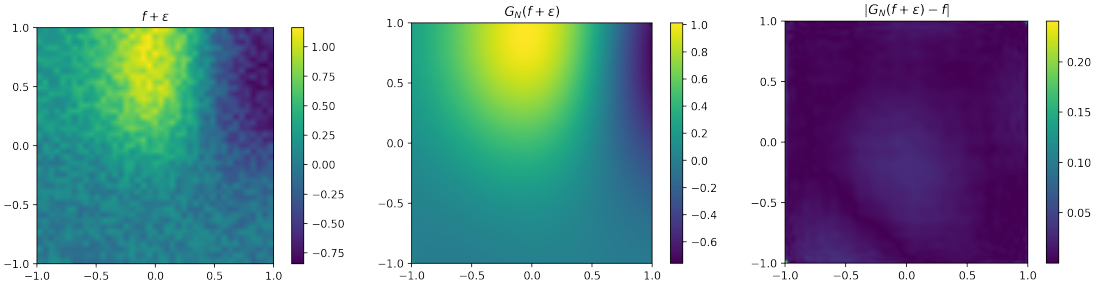

Figure 1.17: **Example of denoising for two-dimensional case. (Left)** $f + \varepsilon$. **(Middle)** $G_N^{f+\varepsilon}$. **(Right)** $|G_N^{f+\varepsilon} - f|$

becomes increasingly challenged to produce accurate results. $\Lambda_N^{\eta,\infty}$ for classical interpolation in Equidistant points increase exponentially and it proves again that equidistant points are not suitable for interpolation.

Another phenomenon that needs attention is that, for G_N and I_N, increasing the number of interpolation points (from $M = (N+1)^2$ to $M = 9(N+1)^2$) enhance model's ability to filter out noise. And the improvement for I_N with Equidistant points is extremely significant. This indicates that Equidistant interpolation points contain sparser information compared to Chebyshev points, and therefore increasing the number of interpolation points (M) can provide more effective information.

For Lagrange polynomials (Figure 1.20), G_N performs stronger ability to filter noise.

1.4 Conclusions

In this chapter, we have introduced the utilization of Neural Networks (NNs) to predict polynomial coefficients, denoted as the operator G_N, and have compared it with traditional polynomial interpolation. Our investigation has considered various scenarios, including Lagrange and Leg-

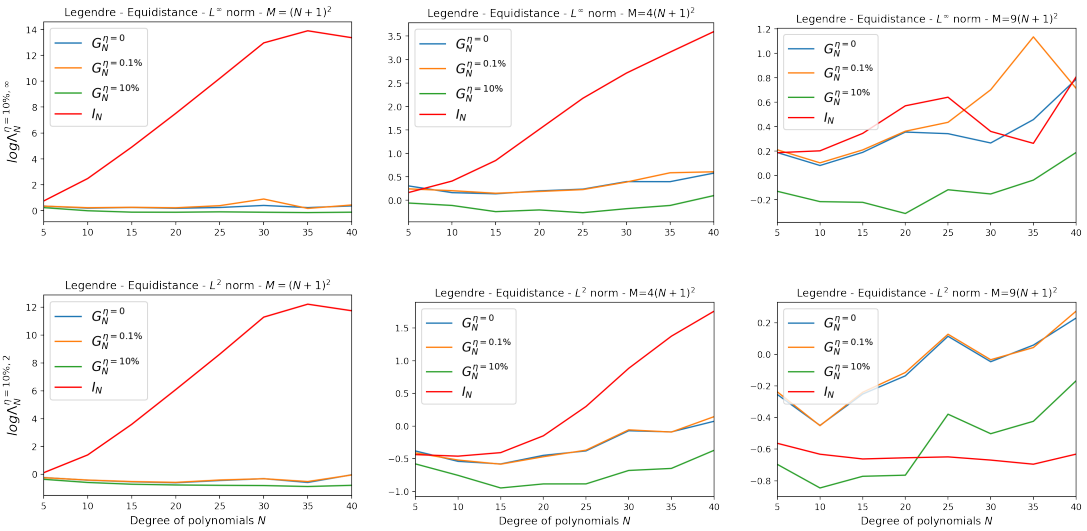

Figure 1.18: $\Lambda_N^{\eta,p}$ **(2D) for Legendre polynomials and Equidistant points. (Left)** $M = N_c$. **(Middle)** $M = 4N_c$. **(Right)** $M = 9N_c$. **(Top)** $\Lambda_N^{\eta,\infty}$. **(Bottom)** $\Lambda_N^{\eta,2}$

endre polynomials, Equidistant and Chebyshev points, varying polynomial degrees, and different levels of noise. Furthermore, we have introduced the Lebesgue constant with perturbation ($\Lambda_N^{\eta,p}$) as a quantitative measure to evaluate the stability of the model when dealing with noisy data.

Our methodology has been validated in both One-dimensional (1D) and Two-dimensional (2D) cases. To reduce model size, we have employed Convolutional Neural Networks (CNNs) instead of fully connected NNs for predicting coefficients in 2D cases. Despite encountering technical challenges related to the precision of floating-point numbers, we have managed to obtain approximation errors and $\Lambda_N^{\eta,p}$ values that suffice to draw the following conclusions:

1. NNs exhibit the capability to effectively capture the characteristics of original functions and approximate them with polynomials, achieving accuracies within the range of 10^{-2} to 10^{-1}, regardless of whether the training data includes noise or not. However, it's important to note that the approximation provided by G_N is not perfect. Even with increased polynomial degree or more complex NN architectures, it cannot significantly improve its approximation capabilities, which we attribute to the inherent limitations of the parameterization task.

2. Despite being a nonlinear operation, G_N can still maintain additivity when employed for polynomial interpolation with surprising performance.

3. Generalization of G_N remains a significant challenge for future research. Presently, NNs struggle to parameterize functions beyond those present in the training data.

3. When we introduce noise filtering tasks during the training phase, G_N demonstrates the ability to effectively filter out noise in the test set. The Lebesgue constant with perturbation, $\Lambda_N^{\eta,p}$, proves that models trained in this manner exhibit stronger noise resistance compared to traditional polynomial interpolation.

4. The Equidistant points should be avoided due to the presence of Runge phenomenon.

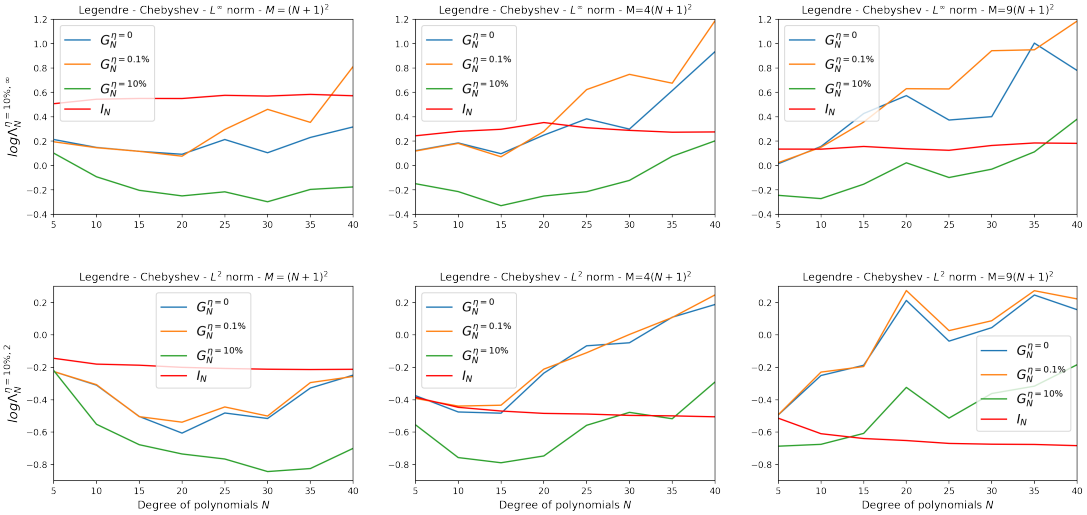

Figure 1.19: $\Lambda_N^{\eta,p}$ **(2D) for Legendre polynomials and Chebyshev points. (Left)** $M = N_c$. **(Middle)** $M = 4N_c$. **(Right)** $M = 9N_c$. **(Top)** $\Lambda_N^{\eta,\infty}$. **(Bottom)** $\Lambda_N^{\eta,2}$

This preliminary work represents an initial exploration of our research. We have found that our models exhibit comparable accuracy to classical interpolation methods in low dimensions. As the dimensionality increases, our model is unable to provide more accurate approximations. This phenomenon is contrary to our initial expectations for ML. We have undertaken several approaches to enhance accuracy, such as increasing the complexity of models, implementing alternative training strategies, and redesigning our loss function. Unfortunately, our approximation functions fail to perform as good as classical interpolation methods.

One potential explanation for our results is that we have not employed an appropriate model for these regression tasks. There should be some special architecture that allows us to fit coefficients. Furthermore, we conjecture that NNs may not be well-suited for tasks involving low-dimensional parameterization. This is because the accuracy of approximation is critically dependent on the coefficients, and even minor variations in these coefficients can have a significant impact. In the context of regression tasks performed by NNs, a certain level of error consistently exists, which becomes especially problematic in cases of low-dimensional parameterization. Our experiments in the last chapter, which focused on force field parameterization, showed that this issue is less evident in high-dimensional problems.

An important future direction of this work is to develop other more flexible models, such as Graph NNs (GNNs), for predicting interpolation coefficients [113]. Unlike fully connected NNs and CNNs, which require structured input data, GNNs operate solely based on nodes and edges, offering extreme flexibility to the model. With GNNs, there is no more need to train separate models for different degrees of polynomials (N). A well-trained GNNs model is supposed to be compatible with any number of input values and consistently yield accurate polynomial coefficients.

Another promising direction is enhancing model's generalization capabilities. A truly versatile and accurate model should be capable of providing suitable approximate functions even when faced with functions it has never encountered during training. This ability to generalize is paramount for neural networks (NNs) to effectively handle new and previously unseen data.

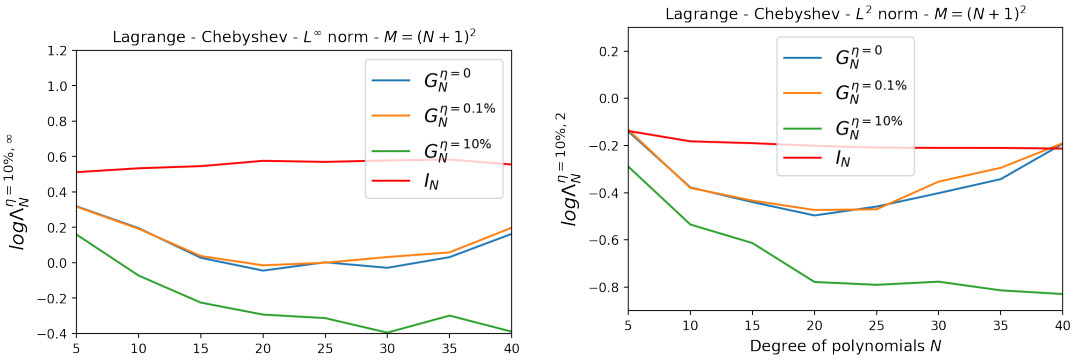

Figure 1.20: $\Lambda_N^{\eta,p}$ **(2D) for Lagrange polynomials and Chebyshev points. (Left)** $\Lambda_N^{\eta,\infty}$. **(Right)** $\Lambda_N^{\eta,2}$.

Techniques like transfer learning can be employed to achieve this level of generalization. By improving the model's generalization, we ensure its competence across a wide range of tasks and its applicability in novel scenarios, rendering it a more robust and dependable tool for practical applications.

Another direction is to enhance the model's generalization ability. We expect the model to be capable of providing suitable approximate functions even when faced with functions it has never encountered during training. This generalization capability is crucial for NNs to effectively deal with new and unseen data. By improving the model's generalization, we ensure its competence across a wide range of tasks and its applicability in novel scenarios, making it a more robust and reliable tool for practical applications.

1.5 Supplementary Information

In this section, we have some tests to assess the generalization of this methodology to other family of functions, seeing whether NNs can always provide good coefficients.

1.5.1 Test Function 1

We consider the following test function:

$$f(x;\mu=[\theta_1,\theta_2,\theta_3]) = \frac{1}{1+\theta_1 x^2} + \theta_2 \sin(\theta_3 x) \tag{1.25}$$

with $\mu \in [10,40]\times[0,3]\times[-3,3]$ and $x \in [-1,1]$.

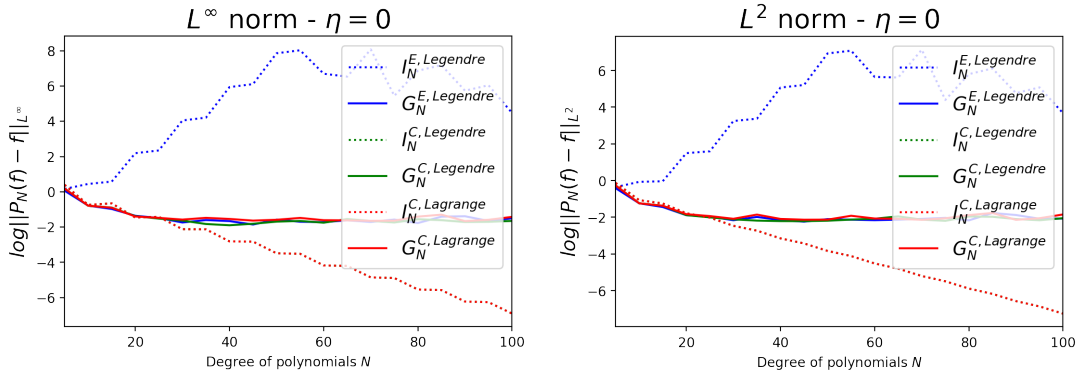

Figure 1.21: **Accuracy of for test function 1.** $\log\|P_N(f)-f\|_{L^p}$ where $p \in \{2,\infty\}$ and P_N can either be classical interpolation I_N (dot line) or neural networks interpolation G_N (full line). **(Left)** L^∞ norm . **(Right)** L^2 norm.

The accuracy are shown in Figure 1.21. We use the same conditions in Section 1.2.3, which means we only consider the case $M = N+1$, the models G_N are trained by noise-free data ($\eta = 0$) and the test data also have no noise. The blue dot lines (classical interpolation for Legendre polynomials and Equidistant points) behave abnormal owing to Runge phenomenon.The examples in Figure 1.22 have also proved it.

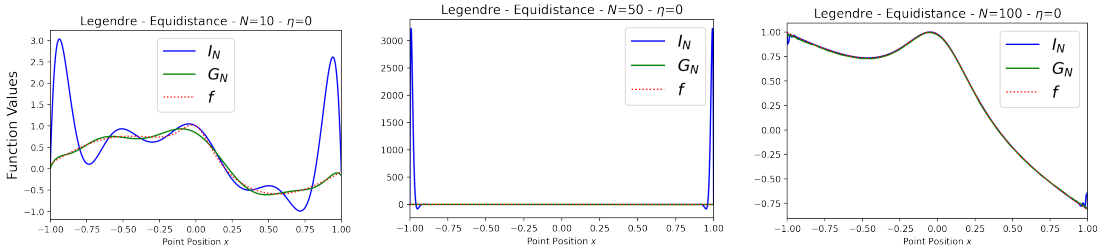

Figure 1.22: **Examples of approximation accuracy for test function 1. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $M = 100$.

There are three examples for functions with maximum approximation error in Figure 1.22. The error for additivity is shown in Figure 1.23. It is also similar to the results in Section 1.2.4. Three examples about the additivity of test functions in Figure 1.24.

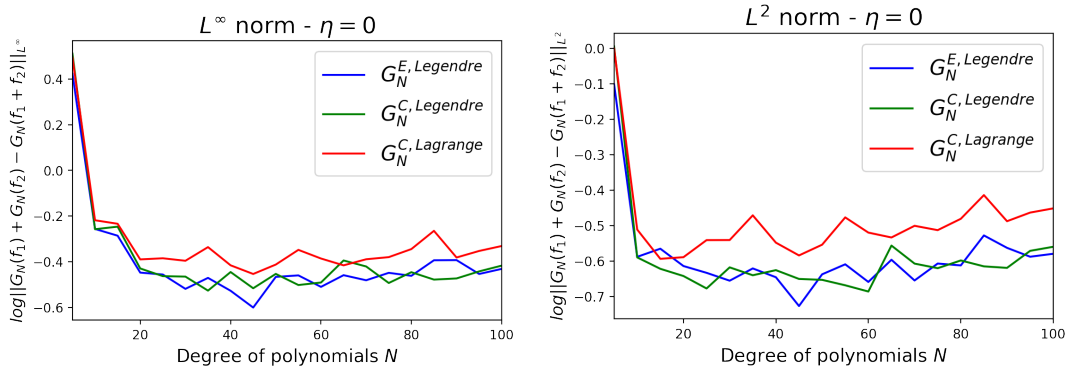

Figure 1.23: **Additivity of G_N for test function 1.** $\log\|G_N(f_1) + G_N(f_2) - G_N(f_1 + f_2)\|_{L^p}$ where $p \in \{2, \infty\}$. **(Left)** L^∞ norm. **(Right)** L^2 norm.

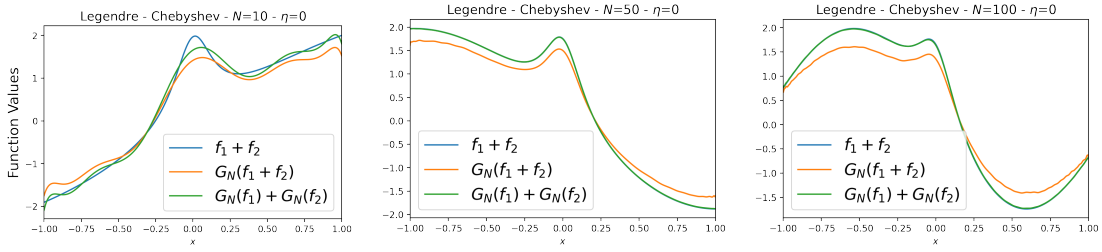

Figure 1.24: **Example of additivity for test function 1. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $N = 100$.

1.5.2 Test Function 2

We consider another test function:

$$f(x; \mu = [\theta_1, \theta_2, \theta_3]) = \sum_{i=0}^{100} (\theta_1 + \theta_2 \exp^{\theta_3 i}) x^i \tag{1.26}$$

with $\mu \in [-10, 10] \times [-10, 10] \times [-1, 1]$ and $x \in [-1, 1]$.

The approximation results and examples are shown in Figure 1.25 and 1.26.

This function is more complicated and have brutal change around $x = 1$. The error for additivity is shown in Figure 1.27. For this family of functions, G_N preserves good additivity.

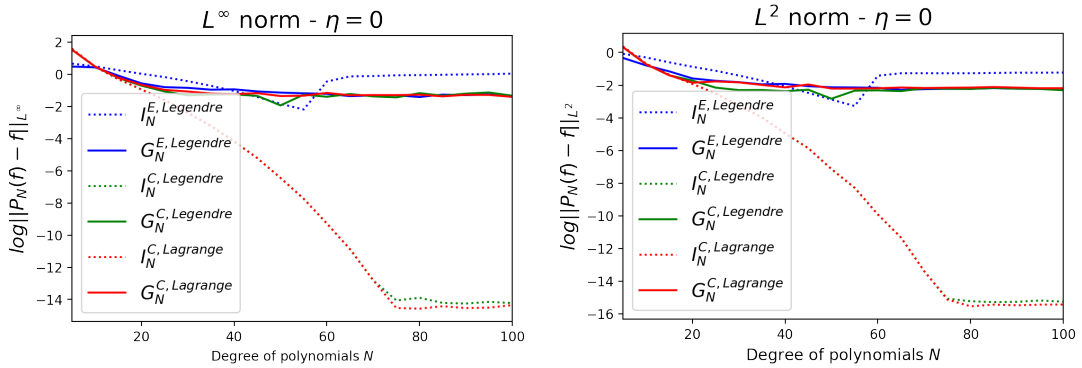

Figure 1.25: **Accuracy of for test function 2.** $\log\|P_N(f) - f\|_{L^p}$ where $p \in \{2, \infty\}$ and P_N can either be classical interpolation I_N (dot line) or neural networks interpolation G_N (full line). **(Left)** L^∞ norm . **(Right)** L^2 norm.

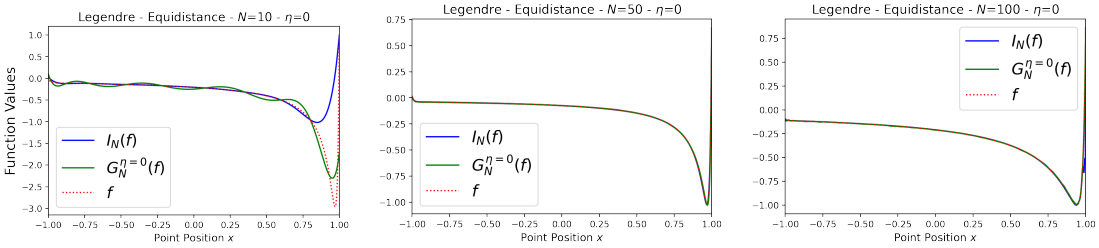

Figure 1.26: **Examples of approximation accuracy for test function 2. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $M = 100$.

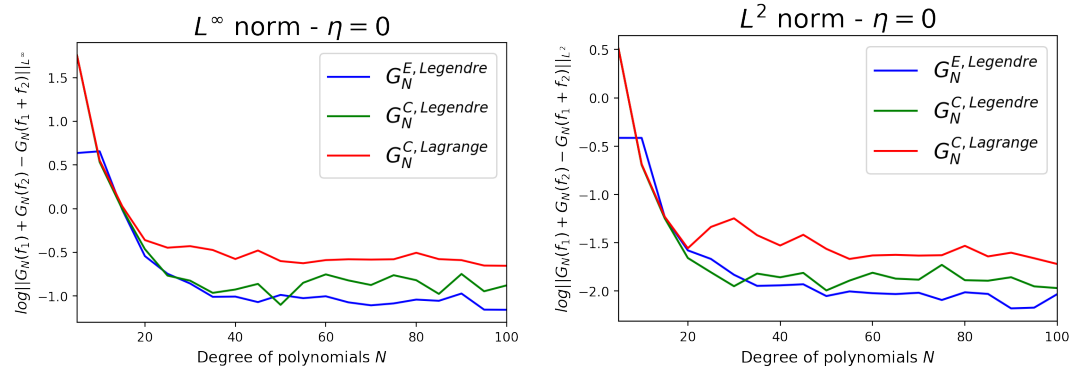

Figure 1.27: **Additivity of G_N for test function 2.** $\log\|G_N(f_1) + G_N(f_2) - G_N(f_1 + f_2)\|_{L^p}$ where $p \in \{2, \infty\}$. **(Left)** L^∞ norm. **(Right)** L^2 norm.

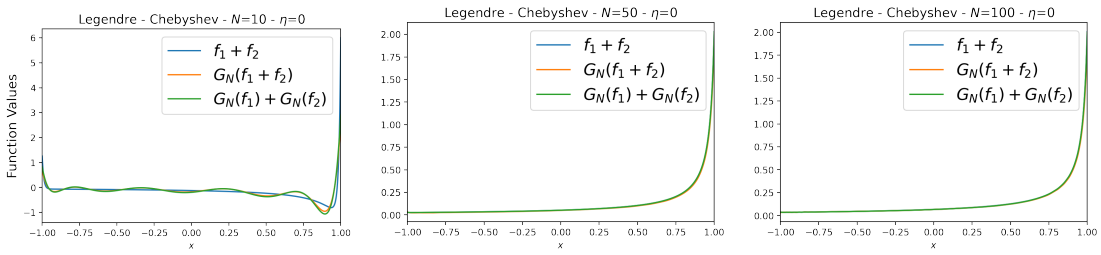

Figure 1.28: **Example of additivity for test function 2. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $N = 100$.

1.5.3 Test on Untrained Functions

We test G_N's ability to approximate functions in untrained families. Considering the combination of Chebyshev points, Legendre polynomials, $M = N + 1$ and $\eta = 0$. The models G_N are trained by Function 1.11 and they will be tested on the test Function 1.25 and 1.26.

As the approximation results are pretty bad, we only present the examples with maximum approximation error in Figure 1.29 (for test function 1.25) and Figure 1.30 (for test function 1.26).

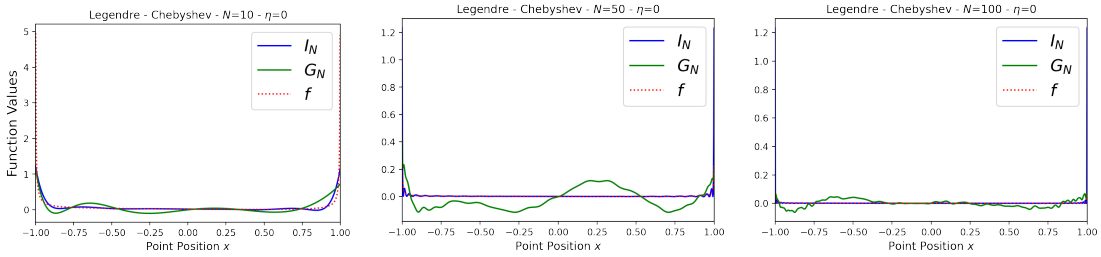

Figure 1.29: **Approximation on untrained test function 1. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $N = 100$.

If the fitted function does not belong to the family of the training dataset, the approximation will be very poor. However, the classical interpolation method I_N (red lines) consistently provide good approximation functions when N is large enough.

NNs fundamentally learn the distribution of the samples and have the ability to interpolate/extrapolate beyond the given data points. However, this capability is not unlimited. When faced with a distribution of function they have never encountered during training, NNs are unable to provide effective predictions.

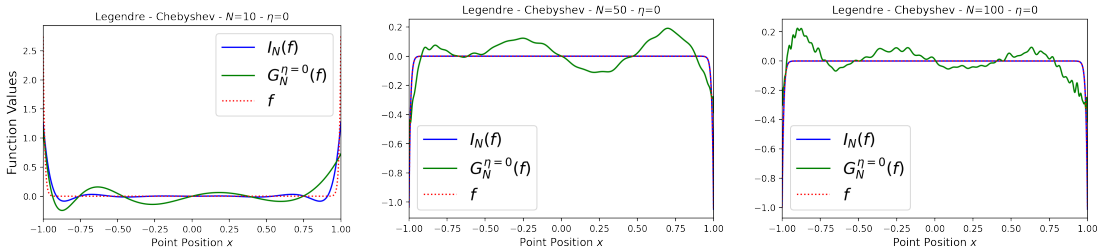

Figure 1.30: **Approximation on untrained test function 2. (Left)** $N = 10$. **(Middle)** $N = 50$. **(Right)** $N = 100$.

Chapter 2

Natural Language Processing (NLP)

Natural Language Processing (NLP) stands at the intersection of linguistics, computer science, and artificial intelligence, with its primary focus on enabling computers to interact with and analyze human language. Specifically, NLP aims to equip computers with the ability to process and analyze large amounts of natural language data. NLP traces back to the 1940s, after World War II, with the increasing need for automated language translation.

Between 1957 and 1970, NLP researchers divided into courants. One focused on rule-based models, exploring formal languages and syntactic, while the other pursued statistical and probabilistic approaches, working on problems like optical character recognition and text pattern recognition.

After 1970, as technology and knowledge expanded, NLP researchers split even further. One new area was logic-based paradigms, languages that focused on encoding rules and language in mathematical logics. This branch eventually contributed to the development of the programming language Prolog. Another area heavily influenced by NLP was natural language understanding, notably shaped by SHRDLU, Professor Terry Winograd's doctoral thesis. From 1983 to 1993, probabilistic and statistical methods dominated NLP, while the past decade has witnessed an evident shift toward information extraction and generation, driven by the vast amounts of information scattered across the Internet.

Nowadays, the objective is to enable computers to comprehend and interpret document contents, including the subtleties of contextual language. This technology facilitates the accurate extraction of insights from documents, their categorization, organization, and even the generation of new text. NLP's applications encompass automatic summarization [114], text generation [115], machine translation [116], text correction [117], etc.

Among the remarkable achievements in NLP, ChatGPT stands out. It is an AI-powered conversational agent developed by OpenAI, based on the GPT-3.5 architecture. It learns from the vast amount of text data available on the internet, making it capable of generating responses that are fluent, coherent, and contextually relevant. ChatGPT holds the potential to be applied in all fields of NLP and to transform the way human beings interact with machines.

In this chapter, we study the application of NLP for processing molecules and predicting molecular properties. This chapter comprises four essential sections:

- **Input and Preprocessing Techniques:** NLP models heavily rely on data, and the quality of the input data significantly impacts their performance. Section 2.1 discusses the representation of molecules as text and the preprocessing of input data.
- **Model Architecture:** The core of NLP lies in the technology used to aggregate and

analyze text. Section 2.2 introduces an efficient and rational algorithm, the attention mechanism.

- **Results:** We assess our NLP model's performance on various benchmark databases, and the outcomes are shown in Section 2.3.
- **Conclusion:** The summary of this chapter is in Section 2.4.

2.1 Chemical Language

NLP focuses on the intricate interaction between computers and human languages, including languages like English, French, and Mandarin, both spoken and written. When applying Machine Learning (ML) techniques to handle molecules, the initial challenge lies in establishing a comprehensible means for computers to describe these molecules effectively. In this section, we will talk about the process of representing molecules using the language of chemistry and discuss the subsequent treatment of these representations.

2.1.1 Simplified Molecular-Input Line-Entry system (SMILES)

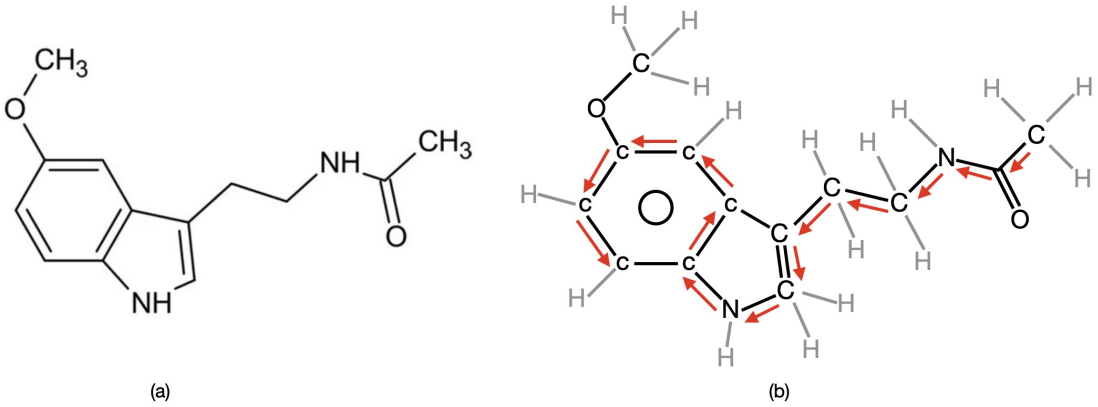

Figure 2.1: **Example of SMILES. (a)** The molecule Melatonin. **(b)** Melatonin expressed in SMILES: CC(=O)NCCC1=CNc2c1cc(OC)cc2 (all hydrogen atoms are ignored). The order of atoms in main chain is indicated in red arrows.

The Simplified Molecular-Input Line-Entry system (SMILES) [72] provides a concise method for representing molecules. SMILES employs a line notation that describes chemical structures in a graph-based framework, encoding atoms, bonds, and rings within text sequences. SMILES can be regarded as a "chemical language" characterized by strings adhering to a regular grammar. For example, consider the molecule Melatonin, as shown in Figure 2.1, expressed as: "CC(=O)NCCC1=CNc2c1cc(OC)cc2". Simply speaking, the letters, e.g., "C", "N", generally represent the atoms and the lowercase, e.g., "c", is an atom in aromatic ring. The numbers represent the start and the end of ring and symbol "=" represents the chemical double bonds. Here are the general grammars in SMILES:

Atoms: In the common case, atoms are represented by the standard abbreviation of the chemical elements without square brackets, such as "C" for carbon and "Br" for bromine. But

brackets may be needed in the following cases: 1.The atom has formal charge (e.g., [Ti+4]). 2. To indicate the hydrogens attached to the atom because atoms' valence is sometimes hard to judge (e.g. for N and P, their valence can be 3 or 5). 3. Atom isotopes (e.g., "H" for hydrogen but "[2H]" for deuterium). 4. The atom is out of the "organic subset (B, C, N, O, P, S, F, Cl, Br, or I)" such as [Au] for gold. 5. Other special cases.

Bonds: Single, double, triple, and quadruple bonds are represented by the symbols "-", "=", "#", and "$" respectively. But the single bond is usually omitted. An additional type of bond is a "non-bond", indicated with ".", to indicate that two parts are not bonded together. For example, aqueous sodium chloride may be written as "[Na+].[Cl-]" to show the dissociation.

Rings: Ring structures are written by breaking each ring at an arbitrary point and the numbers are to indicate the start or the end of ring. For example, cyclohexane and dioxane may be written as C1CCCCC1 and O1CCOCC1 respectively.

Aromaticity: Aromatic rings can be written in three forms, here we only introduce the most common case: writing the constituent "B", "C", "N", "O", "P" and "S" atoms in lower-case forms "b", "c", "n", "o", "p" and "s", respectively. For instance, benzene and furan can be represented respectively by the SMILES "c1ccccc1" and "o1cccc1". Aromatic nitrogen bonded to hydrogen, as found in pyrrole must be represented as "[nH]". Thus imidazole is written in SMILES notation as "n1c[nH]cc1".

Branching: Branches are described with parentheses, as in "CCC(=O)O" for propionic acid. The first atom within the parentheses, and the first atom after the parenthesized group, are both bonded to the same branch point atom.

Stereochemistry: Configuration around double bonds is specified using the characters "/" and "\" to show directional single bonds adjacent to a double bond. For example, "F/C=C/F" is trans-1,2-difluoroethylene whereas "F/C=C\F" is cis-1,2-difluoroethylene.

NLP is a challenging field due to the complexity of natural language, the inherent ambiguity and variability of human communication. Similarly, SMILES faces the difficulty of ambiguity. Vanilla SMILES does not offer a bijective mapping between SMILES sequences and molecules. For example, "CCO", "OCC" and "C(O)C" represent the same molecule (see Figure 2.2). To address this issue, there are two solutions:

- Using canonicalization algorithms to ensure the uniqueness of each molecular structure's representation and establish an one-to-one mapping between SMILES and molecules [118]. This approach aims to standardize SMILES representations, mitigating ambiguity.
- Enabling the model to associate the same molecules with its different SMILES representations. This approach acknowledges the existence of multiple valid representations for the same molecule and develop the model's ability to accommodate such diversity. This is the strategy we have implemented in our model.

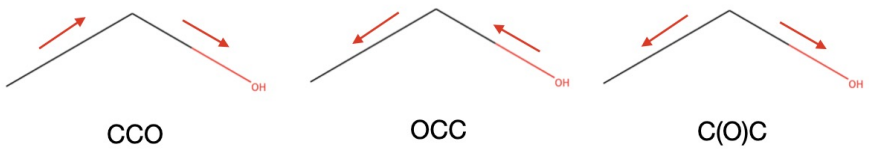

Figure 2.2: **Same molecule be represented by different SMILES**

SMILES can be thought as a language we use in our daily lives and be compatible with nearly all kinds of molecules and ions. It encodes geometric information in molecules, allowing us to uncover their properties solely through analysis of their SMILES representations. This process is similar to analyse the meaning of a sentence for the following reasons:

- SMILES/Sentences are both variable length sequences with strict grammar.
- SMILES/Sentences are composed of characters/words.
- The order of characters/words in SMILES/sentence is important to the outputs. For instance, "John helps Susan" and "Susan helps John" have different meanings. "COC" and "CCO" are two different molecules.

SMILES for one molecule is equivalent to a sentence, with each character, such as in "C=C", corresponding to an atom or a relationship between atoms. Consequently, every character within SMILES can be considered equivalent to a word in a phrase. This is why we employ NLP techniques when working with SMILES.

2.1.2 Tokenized Method

Tokenization is the process of breaking text into smaller units, called tokens. In the context of NLP, tokens typically represent words, although they can also encompass subwords, punctuation marks, or numbers. Tokenization plays a critical role in various NLP tasks as it offers a standard method for representing text data, which can be easily processed by ML models.

In our work, we establish two methods for constructing the vocabulary (i.e., possible characters or tokens) of SMILES:

1. **Functional Groups:** The functional group in the form [...] will be considered as one token. For example, "C[N+]1([O-])COC1" is split into C [N+] 1 ([O-]) C O C 1. Our vocabulary library contains a total of 173 tokens, with undefined tokens represented as [UNK].

2. **Elements:** Functional groups enclosed in [...] are further divided into atoms and numbers. For example, "C[N+]1([O-])COC1" is tokenized as C [N +] 1 ([O -]) C O C 1. This approach substantially reduces the number of possible tokens, resulting in a vocabulary library of only 65 tokens, with undefined tokens still represented as [UNK]. However, it is essential to note that in this context, numbers within SMILES have three distinct meanings:

   - Marking the start and end of a ring, e.g., "Cc1cn[nH]c1"
   - Denoting the charge, e.g., "[Fe+2]"
   - Indicating the number of hydrogen atoms, e.g., "CC(=O)O[AlH3](O)O"

   Thus the numbers have to be carefully classified for different meanings.

In the tokens dictionary, we compile all potential characters from both the pre-training dataset and the test dataset outlined in Section 2.3.1. Tokens that occur infrequently (less than 5 times) are omitted and replaced with [UNK].

2.1.3 One-hot Encoding

One-hot encoding is a widely used technique for representing categorical data, including text data in NLP. In the context of tokenization, it allows us to represent each token as a binary vector, signifying its presence or absence in the text. The following steps outline the process of applying one-hot encoding to input SMILES into an NLP model:

- Establish a vocabulary (refer to Section 2.1.2) that maps each unique token in the text to a distinct integer index, ranging from 1 to 173 or 1 to 65, depending on the tokenization method. For example, the word "C" might be mapped to index 1, "H" to index 2, "c" to index 3, and so on.
- Encode each token in the SMILES as a *one-hot vector*. To achieve this, create a binary vector with a length equal to the vocabulary size. Set the index corresponding to the token's index in the vocabulary to 1, and all other positions to 0. For example, if "C" is at index 1 and the vector should be C=1=(1,0,0,0 ... ,0). Similarly, if "N" is at index 2, the one-hot vector should be N=2=(0,1,0,0 ... ,0).
- Combine these one-hot encoded tokens to form a matrix representation of the entire SMILES. Each row of the matrix represents a single token, and each column corresponds to a unique character in the vocabulary. The values within the matrix are either 0 or 1, indicating the absence or presence of a character in each token.
- Input the matrix representation of SMILES into our NLP model for subsequent processing.

One-hot encoding offers several advantages for NLP tasks:

Simplicity: It is a straightforward and easy-to-implement technique that can be applied to any text data, requiring no prior knowledge of the language or text.

Interpretability: It generates a sparse, binary representation of text data, facilitating the interpretation and visualization of data. This aids in understanding the text's structure and patterns.

Flexibility: It can be employed with various NLP models, including traditional ML algorithms and deep learning models. It can also be combined with other techniques, such as embedding [1], to enhance NLP model performance.

However, one-hot encoding also has limitations:

High Dimensionality: One-hot encoding can lead to a significant increase in the dimensionality of the input space, especially when dealing with large vocabularies or long texts. This may result in computational expenses and challenges in training and optimizing NLP models. Fortunately, in our work, the length of SMILES is closely tied to the size of molecules, and we exclusively consider molecules with hundreds of atoms. As a result, the computational cost remains generally acceptable.

Loss of Context: One-hot encoding treats each token solely, disregarding contextual information about the order or relationships between tokens within the SMILES. This limitation

[1] Embedding is usually a low-dimensional space into which you can translate high-dimensional vectors. An embedding can be learned and reused across models.

can restrict NLP models' capacity to capture intricate linguistic patterns. To address this challenge, we implement a solution known as "positional embedding"[2].

Overall, one-hot encoding proves to be a valuable and commonly employed technique for representing text data in NLP. While it does come with certain limitations, it remains a suitable choice for representing SMILES.

2.2 Introduction to Transformer

In this section, we will introduce the evolution and advancements in NLP technologies. Subsequently, we will present attention mechanisms[49], which serve as the foundation of our NLP model. To enhance the model's performance with unlabeled data, we employ a pre-training and fine-tuning strategy to initialize all neural network parameters.

Here are some notations that will be used in this section.

- D_x is the dimension of input data.
- D_h is the dimension of hidden states.
- D_o is the dimension of output data.
- N is the length of input data.
- t in subscript indicates the number of layers (i.e., position).
- $X = \{x_1, x_2, ..., x_N\}, x_t \in \mathbb{R}^{D_x}$ are inputed node features.
- $H = \{h_1, h_2, ..., h_N\}, h_t \in \mathbb{R}^{D_h}$ are hidden states.
- W and b are trainable parameters in neural networks.
- $\sigma(\cdot)$ denotes a non-linear activation function.
- $\cdot$ is the dot product operation.
- $*$ is the point-wise multiplication operation.

2.2.1 Introduction of Previous Technologies

With the advancement of hardware, the emergence of ML since the 2000s has led to the development of sophisticated NLP models, including deep learning architectures such as Recurrent Neural Networks (RNNs) [119], Long Short-Term Memory (LSTM) [120], Gated Recurrent Units (GRU) [121], and Transformers [49]. These models have the capacity to learn from extensive datasets and have achieved state-of-the-art performance across a broad spectrum of NLP tasks, ranging from machine translation to text classification and sentiment analysis.

Recurrent Neural Networks

Based on David Rumelhart's work in 1986 [119]. Recurrent models typically factor computation along the symbol positions of the input sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input x_t for position t (see Figure 2.3).

[2]Positional embedding assigns unique fixed vectors to each position in a sequence. It aids the model in understanding the order of input data, enabling effective sequence processing without relying solely on the relative positions of tokens.

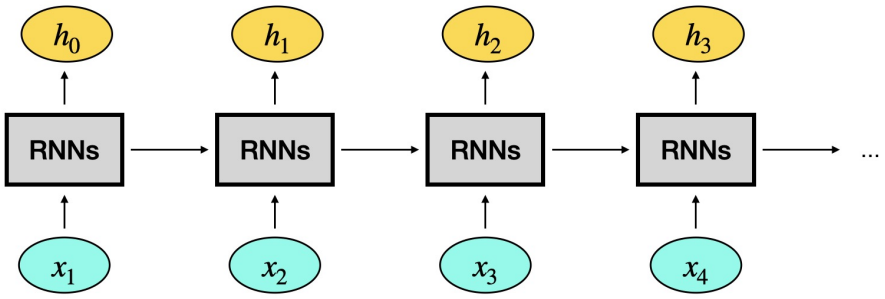

Figure 2.3: **Recurrent Neural Networks Example.** RNNs are recurrent in nature as they perform the same function for every input of data while the output of the current input depends on the past one computation and the current input.

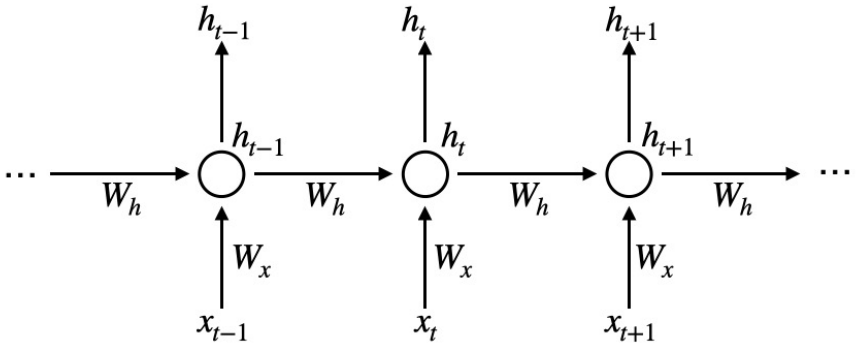

Figure 2.4: **Operation in RNNs.** All RNNs layers share the same parameters W_x and W_h

As shown in Figure 2.4, given the sequential inputs $X = \{x_1, x_2, ..., x_N\}, x_t \in \mathbb{R}^{D_x}$ where N is the length of inputs and D_x is the dimension of inputs. The order of x_t is fixed by SMILES The hidden states h_t are updated by following equations:

$$h_t = tanh(x_t \cdot W_x + h_{t-1} \cdot W_h) \tag{2.1}$$

$H = \{h_1, h_2, ..., h_N\}, h_t \in \mathbb{R}^{D_h}$ are the hidden states, h_N is the final output and D_h is the dimension of hidden states. $W_x \in \mathbb{R}^{D_x \times D_h}, W_h \in \mathbb{R}^{D_h \times D_h}$ are the linear transformation. $tanh$ is a common non-linear activation function defined as:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

According to the propagation algorithm, inputs information is stored in hidden states and h_t is influenced by the inputs from x_1 to x_t.

The inherently sequential nature of RNNs poses limitations on parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batch size across examples. Original RNNs also suffer the vanishing or exploding gradient problem, a common issue in deep learning, due to the same parameters W_x and W_h in all layers (see Figure 2.4). During training via backpropagation from the output layer to the input layer, gradients tend to either shrink or explode. This problem is particularly common in models with too much

layers. Given that the number of layers in RNNs is directly determined by the length of input data, longer sentences can result in excessively deep RNNs.

Long Short-Term Memory

To address the issue of vanishing gradient, LSTM [120] and GRU [121] are invented to avoid the long-term dependency problem. These models have been firmly established as state of the art approaches for the past years in sequence modeling and transduction problems such as language modeling and machine translation [55, 122, 123]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [124–126].

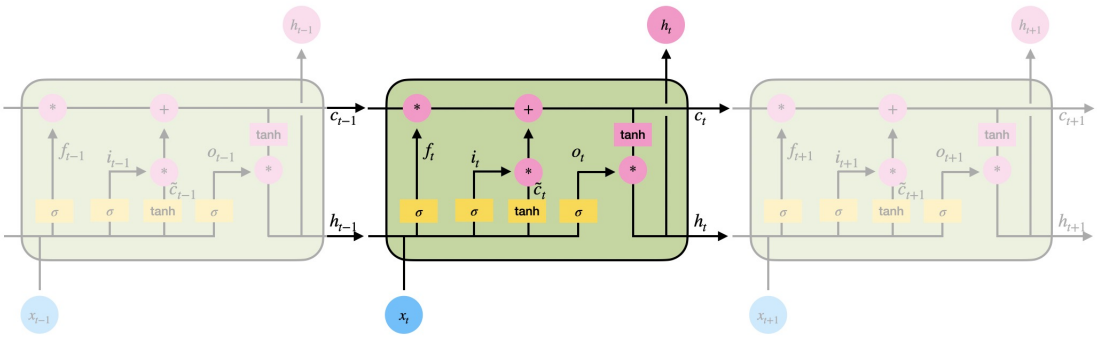

Figure 2.5: **Framework of LSTM**

The following introduction to LSTM is inspired by the blog: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Figure 2.5 shows the framework of LSTM. Its structure is similar to RNNs and every layer share the same parameters. As indicated in Figure 2.6, the key to LSTM is the cell state (noted as c_t), the horizontal line running through the top of the diagram. These new states are used to control the ratio of information through. In the following figures to introduce LSTM, $\sigma(\cdot)$ is the Sigmoid activation function defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, $tanh(\cdot)$ is the activation function defined in Equation (2.2), $\cdot$ is the dot product operation $*$ is the point-wise multiplication operation and $[.,.]$ is the concatenation operation. Besides, we note $x_t \in \mathbb{R}^{D_x}, h_t \in \mathbb{R}^{D_h}, c_t \in \mathbb{R}^{D_h}$ with D_x and D_h are the dimensions of input states and hidden states correspondingly.

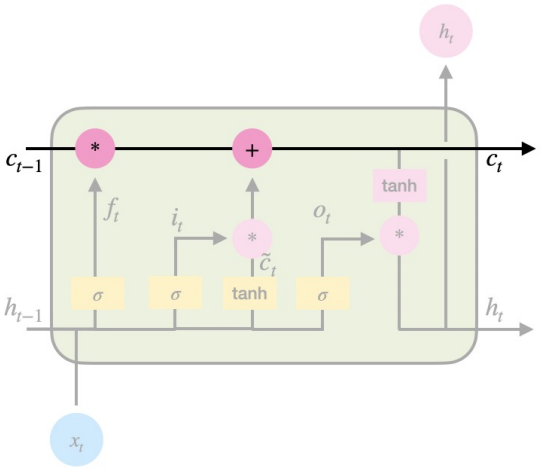

Figure 2.6: **Cell states c_t in LSTM**

There are two gates to update cell states. The first one is called "forget gate layer" showed in Figure 2.7. This is to decide what information we're going to throw away from the cell state. The calculation follows Equation (2.3) with $W_f \in \mathbb{R}^{(D_h+D_x)\times D_h}, b_f, f_t \in \mathbb{R}^{D_h}$. The output f_t is a vector that consists of numbers between 0 and 1, indicating the ratio of the previous cell states to retain.

$$f_t = \sigma([h_{t-1}, x_t] \cdot W_f + b_f) \tag{2.3}$$

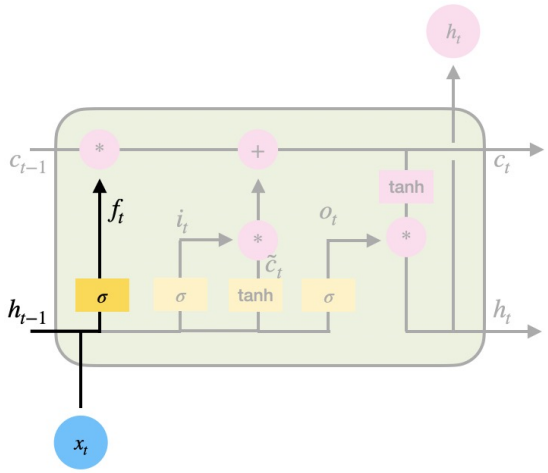

Figure 2.7: **"Forget gate layer" in LSTM**

Another gate to update cell states is "input gate layer", presented in Figure 2.8. This unit decides the new information we're going to add to the cell states. In Equation (2.4), $\tilde{c}_t$ contains the new candidate values that can be added to the cell states. In Equation (2.5), i_t is a vector of number between 0 and 1, indicating the ratio of $\tilde{c}_t$ to be added. $W_c, W_i \in \mathbb{R}^{(D_h+D_x)\times D_h}, b_c, b_i, \tilde{c}_t, i_t \in \mathbb{R}^{D_h}$.

$$\tilde{c}_t = tanh([h_{t-1}, x_t] \cdot W_c + b_c) \tag{2.4}$$

$$i_t = \sigma([h_{t-1}, x_t] \cdot W_i + b_i) \tag{2.5}$$

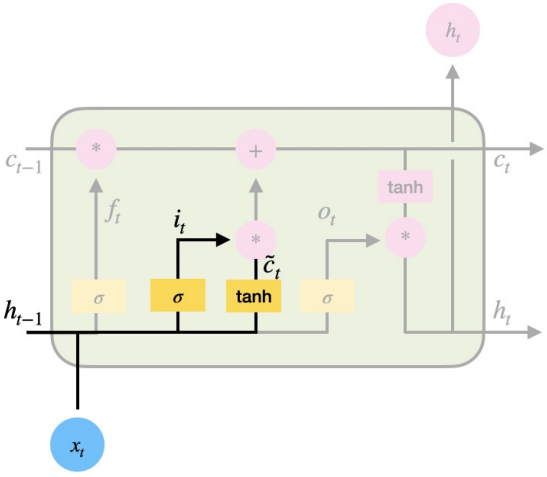

Figure 2.8: **"Input gate layer" in LSTM**

Next step is to update cell states by combining the results of "forget gate layer" and "input gate layer". The operation is in Equation (2.6) and in Figure 2.9.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{2.6}$$

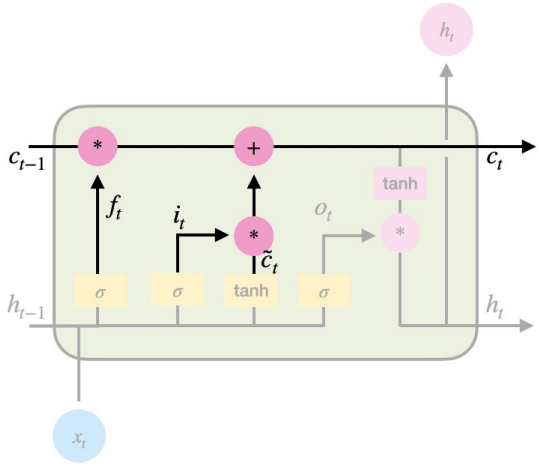

Figure 2.9: **Update of cell states**

Finally, we are going to decide the outputs (To simplify, here we use hidden states as the outputs). Equation (2.7) is the "output gate layer", deciding what parts of the cell states should output. Equation (2.8) shows the update of hidden states. $W_o \in \mathbb{R}^{(D_h+D_x)\times D_h}$, $b_o, o_t \in \mathbb{R}^{D_h}$.

$$o_t = \sigma([h_{t-1}, x_t] \cdot W_o + b_o) \tag{2.7}$$

$$h_t = o_t * tanh(c_t) \tag{2.8}$$

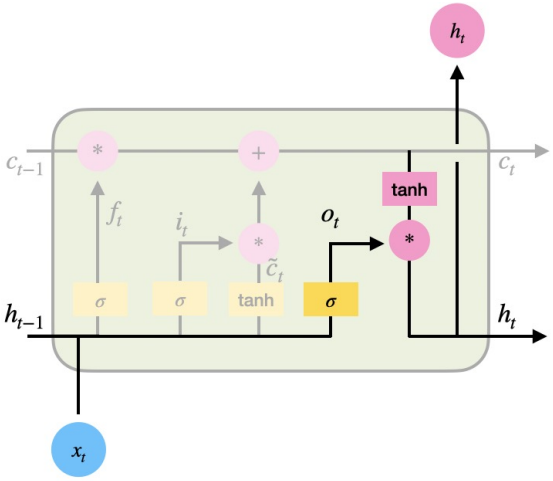

Figure 2.10: **Update of hidden states**

GRU is a simplified variant of LSTM. Thus we do not introduce it in this thesis. Interested readers can refer to article [121] for more details.

While both LSTM and GRU effectively address long-term dependency issues and have demonstrated considerable success in NLP and other sequential data processing tasks, they come with efficiency limitations. Their inherent sequential nature hinders parallelization operations, potentially slowing down training and limiting their efficiency, especially with larger databases.

2.2.2 Self-attention Mechanism

Attention mechanisms have become the most popular of sequence modeling and transduction models in various tasks, allowing the modeling of dependencies without considering their distance in the input or output sequences [55, 127]. The Transformer, introduced in 2017 [49], builds upon these mechanisms. Unlike RNNs, Transformers don't require sequential data to be processed in order. For instance, when processing a natural language sentence, the Transformer doesn't need to handle the beginning before the end. This advantage allows for much greater parallelization compared to RNNs, leading to reduced training times.

Transformer adopts an encoder-decoder architecture, comprising stacked encoder and decoder layers. Encoder is to merge information and to understand the phrase, which is a common part in all NLP tasks. Decoder is to generate a new phrase and it is only necessary in generative tasks like translation and text generation. In this thesus, only encoder is applied.

Encoder layers consist of two sublayers: self-attention followed by a position-wise feed-forward layer. It uses residual connections around each of the sublayers, followed by layer normalization [77], a type of normalization technique. Residual connections are crucial for mitigating the vanishing gradient problem.

It should be noticed that self-attention is a special case of attention. Typically, attention is employed to transfer information from encoder to decoder. However, in the case of self-attention,

it operates within the same component (see Figure 2.11). As this chapter exclusively employs the encoder, unless explicitly specified, any reference to "attention" in the following sections refers to self-attention.

Two common types of attention mechanisms are additive attention [55] and scaled dot-product attention [49]. While their computations are nearly identical, dot-product attention is more space-efficient as it can be implemented using highly optimized matrix multiplication code. Consequently, this thesis exclusively utilizes scaled dot-product attention, denoted as $\cdot$ for dot-product operations.

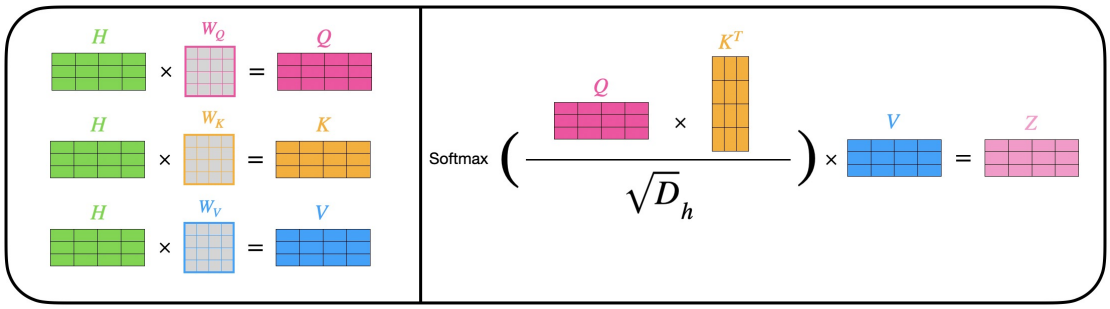

Figure 2.11: **Framework of self-attention mechanism**

The self-attention operates on an input sequence, $H = (h_1, ..., h_N)$ of N characters where for $i = 1, ..., N, h_i \in \mathbb{R}^{D_h}$ are the features for i-th character, and computes a new sequence $Z = (z_1, ..., z_N)$ of the same length where $z_i \in \mathbb{R}^{D_h}$ for $i = 1, ..., N$. D_h is the dimension of input embeddings.

$W_Q, W_K, W_V \in \mathbb{R}^{D_h \times D_h}$ are parameter matrices called query matrice, keys matrice and values matrice. These parameter matrices are unique per layer. e_{ij} is first computed using a compatibility function that compares two input elements:

$$e_{ij} = \frac{(h_i \cdot W_Q) \cdot (h_j \cdot W_K)^T}{\sqrt{D_h}} \tag{2.9}$$

Scaled dot product is chosen for the compatibility function, which enables efficient computation. Linear transformations (i.e., W_Q and W_K) of the inputs add sufficient expressive power.

Then the weight coefficient, α_{ij} , is computed using a `SoftMax` function:

$$\alpha_{ij} = \texttt{Softmax}_j(\{e_{ik}, k = 1, ..., N\}) = \frac{\exp e_{ij}}{\sum_{k=1}^{N} \exp e_{ik}} \tag{2.10}$$

Each output element, z_i, is computed as weighted sum of a linearly transformed input elements:

$$z_i = \sum_{j=1}^{N} \alpha_{ij}(h_j \cdot W_V) \tag{2.11}$$

In summary, combining the Equation (2.9), (2.10) and (2.11), the self-attention operator could be written as:

$$Z = \texttt{Attention}(H, W_Q, W_K, W_V) \tag{2.12}$$

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. Supposing self-attention sublayers employ N_h attention heads, to form the sublayer output, results from each head are concatenated and a parameterized linear transformation is applied. The $\texttt{Attention}(H, W_Q, W_K, W_V)$ in Equation (2.12) is replaced by $\texttt{MultiHead}(H, W_Q, W_K, W_V)$ with:

$$\texttt{MultiHead}(H, W_Q, W_K, W_V) = [head_1, head_2, ..., head_{N_h}] \tag{2.13}$$

where $[., ..., .]$ is the concatenation, $head_i = Attention(H^i, W_Q^i, W_K^i, W_V^i)$.

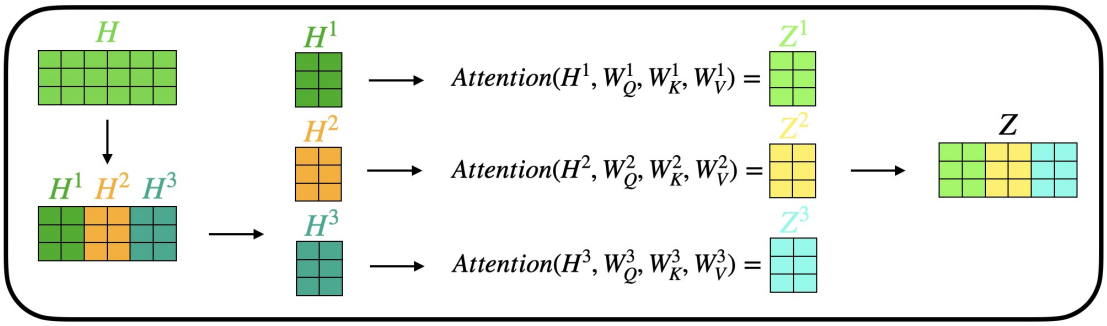

Figure 2.12: **Example of MultiHead attention.** $N_h = 3$

For now, the introduction to attention mechanism is finished. But there is no non-linear activation function from H to Z. Next, we will talk about how to incorporate multi-head self-attention with feed-forward networks and add activation functions to it.

2.2.3 Architecture of Model

We begin with SMILES representations of molecules. Following the two tokenization methods outlined in Section 2.1.2, we divide the SMILES into tokens. Additionally, we insert the tokens [CLS] at the beginning and [END] at the end to signify the start and end of input tokens.

In the output of the Transformer encoder, the token embeddings serve for token-level tasks, such as recovering masked tokens during pre-training. The [CLS] token in the final output plays a role in molecule-level tasks, such as predicting molecular properties. While token embeddings focus on the original token's environment, molecule-level tasks require aggregating information from all tokens. Therefore, we introduce [CLS], a token without specific semantic meaning, to extract global information from the embeddings of all tokens.

Positional embeddings are to indicate the order of input tokens, as elaborated in Section 2.2.2. Since self-attention in Transformers does not inherently consider token order due to parallel processing, we incorporate positional embeddings with token embeddings to capture position information.

As detailed in Section 2.2.2, we've introduced the multi-head self-attention mechanism. Assuming we've passed through t Transformer Encoder layers, we obtain token embeddings denoted as H^t. As shown in Figure 2.13 (b), a skip connection [128] links to the output of multi-head self-attention. Additionally, we apply layer normalization[77] and dropout[129] before passing to the multi layer perception (MLP):

$$Z^{t+1} = \texttt{MultiHead}(H^t, W_Q^{t+1}, W_K^{t+1}, W_V^{t+1}) \tag{2.14}$$

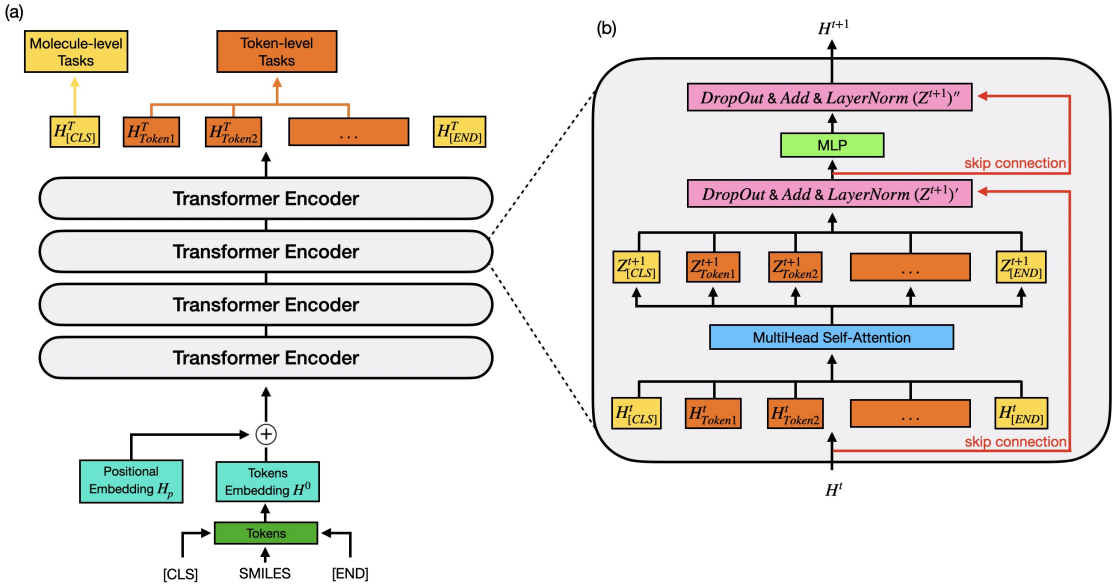

Figure 2.13: **Framework of our NLP model. (a)** It is made up of staked Transformer encoder layers. **(b)** Operation in one transformer encoder

$$(Z^{t+1})' = \texttt{LayerNorm}(H^t + DropOut(Z^{t+1})) \tag{2.15}$$

$(Z^{t+1})'$ will be sent to a MLP with $W_1^{t+1} \in \mathbb{R}^{D_h \times D_{MLP}}, W_2^{t+1} \in \mathbb{R}^{D_{MLP} \times D_h}$ to do linear tranformation, D_{MLP} is the dimension of MLP and $\sigma(\cdot)$ is a non-linear activation function:

$$(Z^{t+1})'' = W_2^{t+1}(\sigma(W_1^{t+1}(Z^{t+1})')) \tag{2.16}$$

We add one more skip connection to $(Z^{t+1})''$ and normalize the result:

$$H^{t+1} = \texttt{LayerNorm}((Z^{t+1})' + \texttt{DropOut}((Z^{t+1})'')) \tag{2.17}$$

Therefore, Equations 2.12-2.17 are the complete operations in one Transformer Encoder.

To accelerate the calculation, we use Rectified Linear Unit (ReLU) activation function as the activation function $\sigma(\cdot)$ instead of $tanh$:

$$\texttt{ReLU}(x) = max(0, x) \tag{2.18}$$

As concluded in [49], Transformer encoder exhibits a computational complexity of $\mathcal{O}(N^2)$. The trainable parameters are initialized with Kaiming initialization [130], an uniform distribution for ReLU activation function.

2.3 Experiments

Our models are implemented in PyTorch [131] frameworks. Following are the details in implementation, including the introduction to test datasets, hyper-parameters in training stage and comparison with other models. And I will explain my reasons for discontinuing my study of NLP

and for not applying these models to my future works.

2.3.1 Databases

ZINC-250K [132] is a well-known molecular database utilized in the experiments to generate molecular graphs. It comprises 249,455 commercially available compounds randomly extracted from the ZINC [133] database. The molecules in ZINC-250K have a maximum of 38 constituent heavy atoms (excluding hydrogen) and consist of various elements, including hydrogen, carbon, nitrogen, oxygen, sulfur, chlorine, fluorine, and more. This database provides essential information, including SMILES, LogP (logarithm of solubility), SAS (synthetic accessibility score), and QED (quantitative estimate of drug-likeness) for each molecule. ZINC-250K is exclusively used for pre-training purposes.

In the fine-tuning process, we select 15 benchmark databases (refer to Table 2.1), covering quantum mechanics tasks (QM7, QM8, QM9), physical chemistry tasks (ESOL, FreeSolv, Lipo), biophysics tasks (PCBA, MUV, HIV, BACE), and physiology tasks (Tox21, ToxCast, ClinTox, BBBP, SIDER) for both classification and regression tasks.

1. **BBBP** The Blood–brain barrier penetration (BBBP) database comes from the study on the modeling and prediction of the barrier permeability [134]. As a membrane separating circulating blood and brain extracellular fluid, the blood–brain barrier blocks most drugs, hormones and neurotransmitters. Thus penetration of the barrier forms a long-standing issue in development of drugs targeting central nervous system. This database includes binary labels for over 2000 compounds on their permeability properties.

2. **SIDER** The Side Effect Resource is a database of marketed drugs and adverse drug reactions [135, 136]. It has grouped drug side-effects into 27 system organ classes following MedDRA classifications [137] measured for 1427 approved drugs.

3. **Tox21** The "Toxicology in the 21st Century" initiative created a public database measuring toxicity of compounds, which has been used in the 2014 Tox21 Data Challenge [6]. This database contains qualitative toxicity measurements for 8014 compounds on 12 different targets, including nuclear receptors and stress response pathways.

4. **ToxCast** It is another data collection [138] (from the same initiative as Tox21) providing toxicology data for a large library of compounds based on in vitro high-throughput screening, including qualitative results of over 600 experiments on 8615 compounds.

5. **ClinTox** The database includes two classification tasks for 1491 drug compounds with known chemical structures: (1) clinical trial toxicity (or absence of toxicity) (2) FDA approval status.

6. **BACE** The BACE database provides quantitative (IC50) and qualitative (binary label) binding results for a set of inhibitors of human β-secretase 1 (BACE-1) [139]. It is used as a classification task for a collection of 1513 compounds.

7. **HIV** Introduced by the Drug Therapeutics Program AIDS Antiviral Screen which tested the ability to inhibit HIV replication for over 40000 compounds [140]. Screening results were evaluated and placed into three categories: confirmed inactive (CI), confirmed active (CA) and confirmed moderately active (CM). We further combine the latter two labels, making it a classification task between inactive (CI) and active (CA and CM).

8. **MUV** Maximum Unbiased Validation group [141] contains 17 challenging tasks for around 90 thousand compounds.

Classification Tasks					
Physiology			Biophysics		
Dataset	Number of Molecules	Number of Tasks	Dataset	Number of Molecules	Number of Tasks
BBBP	2,039	1	BACE	1,513	1
SIDER	1,427	27	HIV	41,127	1
Tox21	7,831	12	MUV	93,087	17
ToxCast	8,575	2	PCBA	437,929	128
ClinTox	1,478	617			
Regression Tasks					
Physical Chemistry			Quantum Mechanics		
Dataset	Number of Molecules	Number of Tasks	Dataset	Number of Molecules	Number of Tasks
ESOL	1,128	1	QM7	6,830	1
FreeSolv	642	1	QM8	21,786	12
Lipo	4,200	1	QM9	133,885	12
ZINC-250K	249,455	3			

Table 2.1: Databases in experiments

9. **PCBA** PubChem BioAssay [142] is a database consisting of biological activities of small molecules, containing 128 bioassays measured over 400 thousand compounds.

10. **ESOL** A small database consisting of water solubility data for 1128 compounds [143]. The database has been used to train models to estimate solubility.

11. **FreeSolv** The Free Solvation Database provides experimental and calculated hydration free energy of small molecules in water [144]. We only use experimental values for comparison.

12. **Lipo** Lipophilicity is an important feature of drug molecules that affects both membrane permeability and solubility. This database provides experimental results of octanol/water distribution coefficient (log D at pH 7.4) of 4200 compounds.

13. **QM7** The QM7 databases [145] is a subsets of the GDB-13 database [146], containing up to 7 "heavy" atoms (C, N, O, S). The electronic properties (atomization energy, HOMO/LUMO eigenvalues, etc.) of each molecule were determined using ab initio density functional theory (PBE0/tier2 basis set). The target is to predict atomization energy directly by chemical structure.

14. **QM8** The QM8 database comes from a study on modeling quantum mechanical calculations of electronic spectra and excited state energy of small molecules [147]. Multiple methods, including time-dependent density functional theories (TDDFT) and second-order approximate coupled-cluster (CC2), are applied to a collection of molecules that include up to eight heavy atoms (also a subset of the GDB-17 database [148]). In total, four excited state properties are calculated by three different methods on 22 thousand samples.

15. **QM9** QM9 is a comprehensive database that provides geometric, energetic, electronic and thermodynamic properties for a subset of GDB-17 database [148], comprising 134 thousand stable organic molecules with up to 9 heavy atoms [149].

All the databases are scaffold split [150] to increase the challenge for learning algorithms. Scaffold splitting is a computational approach to break down a complex chemical structure into its constituent molecular fragments or scaffolds. The goal is to identify and isolate the key substructures or scaffolds that contribute to the overall chemical properties and biological activity of the molecule. The process of scaffold splitting involves identifying and extracting the core framework or backbone of a molecule and the molecules with similar backbone will be assigned to the same set. Therefore the molecules in training set and test set are quite different and models' performance on scaffold splitting sets are always worse than that on random splitting sets. The ratio of training, validation and test sets is 8:1:1.

2.3.2 Metrics

The measurements in the databases can be quantitative or qualitative and we adopted different metrics to compare with previous baselines.

For classification models, we employ the area under the receiver operating characteristic curve (ROC-AUC). It is a performance metric for binary classification problems, quantifying a model's ability to distinguish between positive and negative classes. ROC-AUC measures the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. The score ranges from 0 to 1, where higher values indicate superior model performance. An ROC-AUC score of 0.5 implies no better performance than random guessing, while a score of 1.0 signifies perfect classification.

In regression problems, we assess model performance using mean absolute error (MAE) or root-mean-square error (RMSE). Lower scores in these metrics indicate better model performance.

2.3.3 Pre-Training

In our work, some of the benchmark databases consist of only thousands of molecules. To enhance model performance on such databases, we employ pre-training [151]. Pre-training is a ML technique that involves training a model on a large and diverse dataset before fine-tuning it for a specific task. The primary advantage of pre-training is that it allows the model to learn general features and patterns, which leads to improved performance on the target task. This is particularly beneficial in deep learning models with a large number of parameters, as it helps prevent overfitting when training on small datasets.

There are two established strategies for applying pre-trained language representations to downstream tasks: the feature-based approach and the fine-tuning approach. The feature-based approach, exemplified by ELMo [152], integrates pre-trained representations as additional features into task-specific architectures. In contrast, the fine-tuning approach, as seen in models like the Generative Pre-trained Transformer (OpenAI GPT) [153], introduces minimal task-specific parameters and fine-tunes all pre-trained parameters on the downstream tasks. Both approaches share the same objective function during pre-training.

Before the advent of Bidirectional Encoder Representations from Transformers (BERT), language model pre-training had already demonstrated its effectiveness in improving various natural language processing tasks [152–155]. These tasks encompass sentence-level challenges, such as natural language inference [155, 156], which involves predicting relationships between sentences, as well as token-level tasks like named entity recognition and question answering, where models produce fine-grained token-level outputs [157, 158].

The major limitation for the past pre-training models before BERT was that standard language models were unidirectional, and this limited the choice of architectures that could be used

during pre-training. BERT was specifically designed to address this limitation by pre-training deep bidirectional representations from unlabeled text. Consequently, the pre-trained BERT model can be fine-tuned with just one additional output layer, enabling the creation of state-of-the-art models for a wide range of tasks, including question answering and language inference, without requiring substantial task-specific architecture modifications.

BERT overcomes unidirectionality through the use of a Masked Language Model (MLM) pre-training task, inspired by the Cloze task [159]. In the MLM task, a random percentage of input tokens is masked, and the goal is to predict the original vocabulary id of the masked word solely based on its context. Unlike left-to-right language model pre-training, the MLM objective allows representations to incorporate both left and right context, enabling pre-training a deep bidirectional Transformer.

In addition to MLM, BERT uses the Next Sentence Prediction (NSP) task to capture the relationship between two sentences. Building upon BERT and the Robustly Optimized BERT Approach (RoBERTa) [160], an optimized version of BERT, researchers have applied pre-training models to modern computational chemistry [161–164].

To our NLP models, we have designed three tasks to help the model to better understand SMILES:

Masking Task: Inspired by BERT, we introduce a masking task where 20% of the input tokens are selected for potential replacement. Among the selected tokens, 80% are replaced with [MASK], 10% remain unchanged, and 10% are substituted with randomly generated tokens. The model's objective is to predict the original tokens from the masked inputs (refer to Figure 2.14). After processing through our model, we can extract the vectors corresponding to the masked tokens. These vectors are then passed through a fully-connected layer with non-linear activation to generate a distribution over the model's vocabulary. Training is performed using a cross entropy loss function.

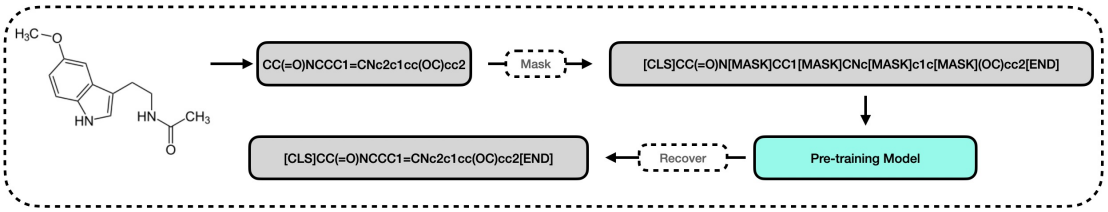

Figure 2.14: **Masking Task.** After splitting the SMILES into tokens and adding [CLS] and [END], some tokens are replaced by [MASK]. The outputs of pre-training model should be able to recover the masked parts.

Same Molecule Classification (SMC): Same Molecule Classification (SMC): Since a molecule can be represented by different SMILES, we aim to train a model that understands relationships between SMILES. The pre-training model collects the [CLS] vectors from various sequence data and determines whether the pairs represent the same molecule (refer to Figure 2.15). This task involves binary classification, with the targets being boolean values (True or False). Evaluation is based on binary cross entropy loss.

Connection classification (CC): In SMILES notation, the presence of parentheses indicates branched chains, making it challenging to identify connected atoms. This task encourages the pre-training model to comprehend these connections (refer to Figure 2.16). The targets are boolean values (True or False), and optimization is carried out using a binary cross entropy loss function.

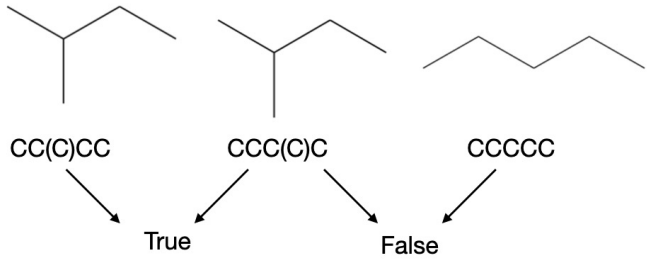

Figure 2.15: **Same Molecule Classification (SMC).** The different SMILES for the same molecule will return "True" while the result of different molecules is "False".

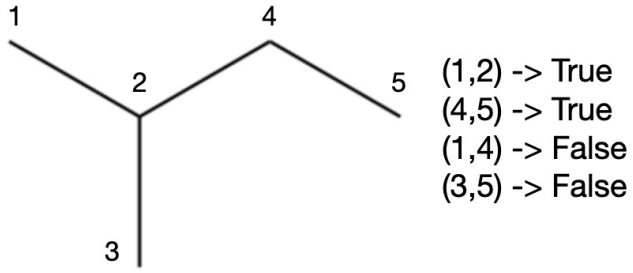

Figure 2.16: **Connection Classification (CC).** The numbers are the indices for atoms, e.g., (1,2) represents the pair of atom 1 and atom 2. The connected atom pairs will return True while the unconnected atom pairs return False.

SMC classification and CC classification are both binary classification tasks. The target labels are 1 (True) or 0 (False) and our prediction result would be a probability between 0 and 1. We take `torch.nn.BCELoss()` function in PyTorch to measure the Binary Cross Entropy (BCE) between the input probabilities and the targets:

$$\texttt{BCELoss}(p, y) = -(y \cdot \log p + (1 - y) \cdot \log(1 - p)) \tag{2.19}$$

where y is the true binary label (0 or 1), p is the predicted probability (between 0 and 1) that the input belongs to the positive class (usually labeled as 1), and log denotes the natural logarithm.

The Masking Task is essentially a multi-class classification problem in which the potential targets correspond to the token dictionary. The objective is to correctly assign the masked tokens to their respective classes. To accomplish this, we employ the torch.nn.CrossEntropyLoss() function provided by PyTorch. Cross entropy loss is utilized to quantify the dissimilarity between the predicted probability distribution and the true probability distribution of the target labels.

$$\texttt{CrossEntropyLoss}(p, y) = -\sum_{c=1}^{C} y_c \log p_c \tag{2.20}$$

where C is the number of classes, $p = (p_1, p_2, ..., p_C)$ is the predicted probability distribution of the target labels and $y = (y_1, y_2, ..., y_C)$ is the true probability distribution of the target labels. Usually, there is only a single positive label (labeled as 1) in y while the rest should be negative label (labeled as 0).

The molecular pre-training dataset comprises all the public databases used for model verification, with the exception of the PCBA database, from which we randomly selected 40,000 molecules. Additionally, we included data from the ZINC-250K database. The inputs for our NLP models consist of SMILES representations (excluding explicit hydrogens). We applied the two tokenization methods discussed in Section 2.1.2 to split the SMILES and filtered out sequences that were either too long (more than 500 tokens) or too short (less than 10 tokens).

The pre-training procedure begins with reading SMILES, followed by random modifications to generate diverse SMILES representations for the same molecule. These SMILES are then tokenized, with the special classification tokens [CLS] and [END] added at the beginning and end, respectively. To encode positional information, we incorporate learned positional embeddings [165] before passing the sequence through the self-attention layers of the model.

Number of layers	4	Dimension of model	512	Initial learning rate	5E-04	Training data	4.9E5
Number of heads	16	Dimension of MLP	1,024	Smallest learning rate	1E-06	Validation data	6.1E4
Functional group tokens	173	Dropout	0.1	Gamma	0.999	Test data	6.1E4
Element tokens	65	Minimum length	10	Maximum length	500	Batch size	Flexible

Table 2.2: Hyper-parameters for NLP pre-training model

Each model is pre-trained for 150 epochs using 1 NVIDIA V100 GPU of 16GB. Pre-training takes approximately 13 hours. As our model is not big (about 38 MB), there is no warmup before pre-training. Initial learning rate for all parameters is $5E-4$, that decays by gamma (see Table 2.2) every 100 steps. The decay will be stopped after learning rate is less than smallest learning rate. Additionally, we use the Adam optimiser [111] with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both pre-training and fine-tuning on all tasks. As larger batch size requires more GPU memory, to fully use the computational power of GPU, the batch size changes with the length of SMILES (see Table 2.3).

Number of Tokens	$N \leq 50$	$50 < N \leq 100$	$100 < N \leq 200$	$200 < N \leq 300$	$300 < N \leq 400$	$N > 400$
Batch Size	512	256	64	32	16	8

Table 2.3: Batch size for NLP pre-training model

Our model is made up of 4 Transformer encoder layers ($T = 4$) with 16 attention heads ($N_h = 16$) at each layer. Dimension of model (D_h in section 2.2.3) is 512 and dimension of feed-forward networks is 1024 (D_{MLP} in section 2.2.3). `DropOut` is set to 0.1 to alleviate overfitting.

Task / Tokenized Methods	Masking	Connection Classification	Same Molecule Classification
Functional Group	93.1%	99.8%	98.8%
Element	94.8%	99.8%	98.6%

Table 2.4: Accuracy of NLP pre-training model

The pre-training results, as presented in Table 2.4, reveal that our models successfully recover masked information and correctly classify atoms and molecules in most cases. Here are some key observations and analyses:

- The functional group tokenization method yields a larger vocabulary dictionary, and each token (functional group) carries more information compared to an element. Consequently,

it is reasonable to anticipate that recovering masked tokens might be more challenging in this context.

- Both tokenization methods exhibit strong performance in recognizing the connections between tokens. This suggests that our NLP models effectively understand SMILES notation and can extract two-dimensional structural information.

- The accuracy of SMC is notably affected by masked and randomly generated tokens. For instance, masking the last character in both "CCO" and "CCC", results in the same expression "CC[MASK]". Since the accuracy of recovering masked tokens is below 95%, these incorrectly predicted tokens impact the extraction of molecular fingerprints, consequently reducing the accuracy of SMC.

2.3.4 Experimental Results

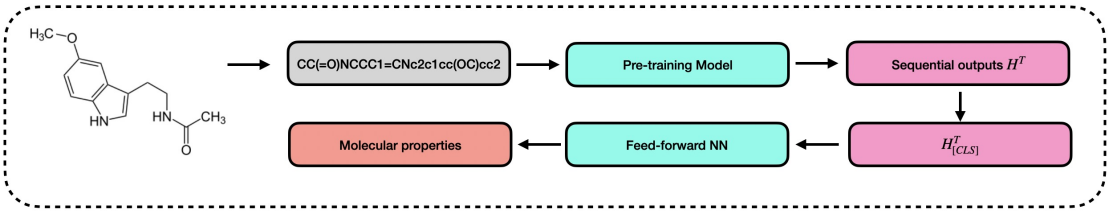

Figure 2.17: **An illustration of the fine-tuning procedures for downstream tasks**

After the pre-training phase, our model is fine-tuned on downstream datasets for the prediction of molecular properties. The inputs are first processed through the encoders, and we extract the embeddings corresponding to [CLS] at the final layer T. These selected fixed-length vectors serve as molecular fingerprints and are then passed through a feed-forward NN (to simplify, we only apply a linear transformation) to make predictions on molecular properties.

For classification tasks, we use the BCE loss (presented in Equation (2.19)) because the databases in Table 2.5 are all binary classifications. For regression tasks, we use MSE loss (see Equation (1.15)).

Classification Tasks ROC-AUC % (higher is better) in form Average (Standard Deviatioin)

Dataset	**BBBP**	**SIDER**	**Tox21**	**ToxCast**	**ClinTox**	**BACE**	**HIV**	**MUV**	**PCBA**	**Avg**
Number of molecules	**2,039**	**1,427**	**7,831**	**8,575**	**1,478**	**1,513**	**41,127**	**93,087**	**437,929**	
Number of prediction tasks	**1**	**27**	**12**	**617**	**2**	**1**	**1**	**17**	**128**	
GROVER_{large}	67.8(0.2)	62.2(1.9)	73.5(0.1)	65.3(0.1)	78.8(0.7)	81.4(1.3)	72.8(0.5)	67.8(1.3)	83.1(0.5)	72.5
AttentiveFP	65.2(0.9)	60.7(1.6)	76.7(0.5)	**67.4(0.1)**	82.8(0.6)	80.9(0.3)	75.4(0.9)	73.0(0.5)	80.3(0.6)	73.6
D-MPNN	**71.2(0.3)**	60.2(0.4)	75.1(0.2)	64.3(0.4)	**89.6(0.2)**	80.0(0.2)	76.4(1.4)	75.3(1.8)	86.2(0.3)	75.4
Pretrain-GCN	70.5(0.9)	62.5(0.2)	75.8(0.3)	65.4(0.1)	63.5(1.8)	84.1(0.4)	76.9(0.7)	79.6(0.2)	84.7(0.1)	73.7
Pretrain-GIN	70.4(0.3)	**62.9(0.1)**	**78.1(0.5)**	65.7(0.5)	73.1(1.2)	**84.4(0.2)**	**79.6(0.1)**	**82.0(0.1)**	**86.5(0.1)**	**75.9**
NLP model (Functional Group)	64.7(1.4)	62.8(0.5)	73.2(0.3)	63.9(0.4)	82.4(0.5)	82.9(1.1)	74.5(0.5)	73.0(0.7)	83.9(0.2)	73.5
NLP model (Element)	68.1(0.6)	60.1(0.6)	74.4(0.1)	62.6(0.3)	84.5(0.2)	74.8(0.3)	76.5(0.2)	76.8(0.4)	84.1(0.1)	73.5

Table 2.5: Classification results for NLP models.

As suggested by the MoleculeNet [78], the mean and standard deviation of the results for three random seeds are listed in Table 2.5 and Table 2.6. We compare our models with five graph-based models, including supervised and pre-training baselines. D-MPNN [52] and AttentiveFP [166] are supervised GNNs methods. Pretrain-GIN and Pretrain-GCN [162], GROVER [34] are pre-training methods. The best results are marked in bold.

Regression Tasks (lower is better) in form Average (Standard Deviation)

	RMSE			MAE		
Dataset	**ESOL**	**FreeSolv**	**Lipo**	**QM7**	**QM8**	**QM9**
Number of molecules	**1,128**	**642**	**4,200**	**6,830**	**21,786**	**133,885**
Number of prediction tasks	**1**	**1**	**1**	**1**	**12**	**12**
$GROVER_{large}$	**0.907(0.002)**	2.888(0.014)	0.817(0.015)	92.4(6.7)	0.0226(0.0017)	5.836(0.111)
AttentiveFP	0.915(0.022)	**1.945(0.094)**	0.729(0.011)	**69.2(2.5)**	0.0181(0.0001)	**4.166(0.146)**
D-MPNN	1.075(0.005)	2.140(0.070)	**0.688(0.009)**	97.6(1.5)	**0.0179(0.0004)**	6.240(0.287)
Pretrain-GCN	1.255(0.014)	2.095(0.114)	0.770(0.005)	83.8(2.2)	0.0200(0.0001)	8.006(0.066)
Pretrain-GIN	1.150(0.023)	2.763(0.075)	0.759(0.012)	94.1(3.8)	0.0201(0.0005)	8.450(0.112)
NLP model (Functional Group)	1.023(0.028)	2.803(0.042)	0.775(0.011)	84.5(1.8)	0.0203(0.0004)	4.314(0.087)
NLP model (Element)	1.204(0.031)	3.282(0.103)	0.821(0.009)	104.9(1.6)	0.0214(0.0002)	4.431(0.128)

Table 2.6: Regression Results for NLP models

Our NLP models have shown their ability to predict molecular properties, demonstrating the potential of attention mechanism in aggregating information and NLP's ability to treat molecules. For classification tasks, the two tokenization methods have same performance. For regression tasks, functional group tokens give smaller error. However, for our models based on these two splitting methods, they are always worse than graph-based models. Compared to SMILES notation, molecular graphs contain greater amount of structural information, which enables more comprehensive fusion to generate molecular fingerprints.

Although we could employ additional techniques to enhance the performance of our NLP models, we stop our researches of NLP models here. We have found that NLP models are naturally not appropriate for predicting molecular force field parameters, which is the most important part of our work.

In traditional force fields, the parameters are decided by atoms (electrostatic and Van der Waals interactions) or atom pairs (stretch interaction) or atom groups (bending and torsions interactions). Therefore, the parameterization of force field is actually an atom-level task. To SMILES, the interaction of multiple atoms is ambiguous and the structural information cannot be well-utilized. For instance, if we expect to decide the parameters for the bending term H-C-H in ethane (the three atoms in red box of Figure 2.18) through NLP model, we have to split SMILES with explicit hydrogen atoms (Usually, the hydrogens in SMILES are ignored. But they have to be considered in force field parameterization tasks.) by elements and choose the corresponding feature vectors to the three red characters in Figure 2.18(b). The structural information in SMILES is stored in special characters, e.g., "(" or "[", which do not participate the parameters generation. In contrast, if we treat the ethane as a molecular graph, we choose the corresponding atoms and edges (the red parts in Figure 2.18(c)).

According to our experiments on force field parameterization, the errors are too large to be acceptable. The atom-level task is different from the character-level task and the three pre-training tasks (Masking task, SMC and CC) fail to inspire models to discover the existence of angle, let alone torsion structure. Furthermore, SMILES is unclear about the long range interaction, like Van der Waals interaction.

2.4 Conclusions

In this chapter, we explored the representation of molecules as text strings (SMILES) and applied Natural Language Processing (NLP) models to extract molecular fingerprints. SMILES is an expressive chemical language, with atoms and bonds represented as characters (Section 2.1.1). We adopted two distinct methods to tokenize SMILES: by elemental characters or by functional groups (Section 2.1.2). Interestingly, our experiments revealed that the choice of tokenization

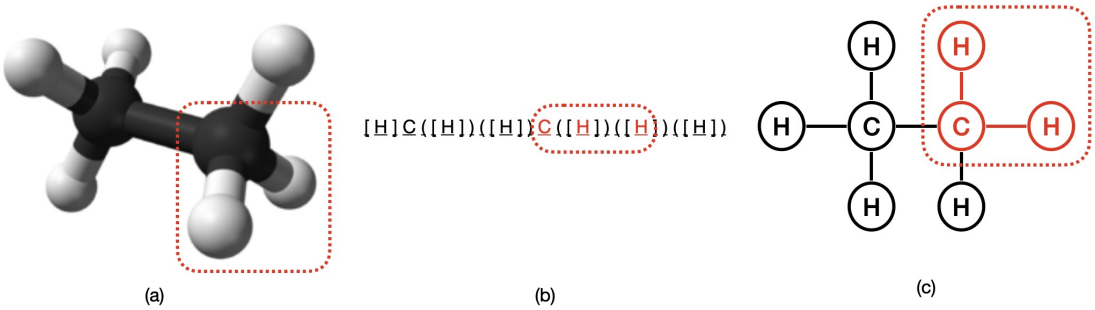

Figure 2.18: **The same angle term in Ethane. (a)** Ball-and-stick model. **(b)** SMILES split by element. **(c)** Molecular graph

method had no significant impact on the fine-tuning results (Table 2.5 and 2.6).

Before the appearance of the Transformer model, processing sequential data bidirectionally was a challenge. Traditional Recurrent Neural Networks (RNNs) suffered from inefficiency due to their inherent structure. This limitation came from the natural architecture of RNNs and was difficult to overcome. That is why we focus on the self-attention mechanism, which allows for much more parallelization. The introduction of the Transformer encoder layer, based on the multi-head self-attention mechanism, greatly accelerated training process and made it feasible to collect extremely large datasets for pre-training.

The core structure of our model consists of stacked Transformer encoder layers, as illustrated in Figure 2.13. While a greater number of layers theoretically allows for improved information fusion, deeper models can suffer from over-smoothing issues. Therefore, determining the optimal number of layers is a critical hyper-parameter that requires careful consideration. A Transformer encoder layer comprises a multi-head self-attention operation and two-layers feed-forward networks, which introduce non-linearity into the model while preserving skip connections.

In order to make use of the unlabeled data and to alleviate the possible over-fitting problem, we pre-trained a four-layers Transformer encoders by three unsupervised tasks: Masking task, Same Molecule Classification and Connection Classification. The pre-training process took approximately 13 hours. The fingerprints generated by the pre-trained model proved to be directly applicable to downstream tasks. During pre-training, our model showcased exceptional capabilities, achieving outstanding accuracy in distinguishing link relations (99.8%) and recognizing the same molecule represented by different SMILES (98.6%). However, recovering masked tokens posed a more significant challenge, with an accuracy of less than 95%. This is attributed to the intricacies of SMILES grammar and the diversity of possible compounds.

The results presented in Table 2.5 and Table 2.6 prove that NLP models are indeed effective tools for predicting molecular properties. This holds significant promise for applications such as drug design and molecule discovery. It should be noted that our models used in this chapter are only to reproduce the existing NLP models and they do not outperform any other graph-based baselines on any benchmarks. This does not mean that NLP models are definitely inferior to graph neural networks because with the development of NLP, especially with the application of ChatGPT, NLP has showed its potential in all kinds of domains, even the computer vision and autopilot where we thought unrelated to NLP. Nevertheless, the success of ChatGPT is due to its 175 billions parameters and the infinite high quality text on Internet. It is almost impossible to reproduce a similar machine learning model in chemistry domain.

Moreover, applying NLP models to derive force field parameters encounters inherent challenges, as explained at the end of Section 2.3.4. SMILES characters represent sequential data,

making it complex to extract molecular geometry information. While we made an attempt to apply NLP for force field parameter prediction, the relative error sometimes exceeded 10%, far surpassing an acceptable margin.

After realizing that NLP might not be the ideal approach for predicting force field parameters, we redirected our efforts toward Graph Neural Networks (GNNs). GNNs are more flexible and are strong at processing the unstructured data because they take the nodes and edges as inputs. By treating the atoms as nodes and the bonds as edges, molecules are easily represented as graphs (see Figure (c)). Even with relatively simple graph-based aggregation algorithms, small models like Pretrained GIN (7.5 MB) [162] outperformed our NLP model (38 MB) in certain scenarios.

In summary, NLP models have proven effective for analyzing molecular properties, but predicting force field parameters appears to be a challenge beyond their current capabilities.

Chapter 3

Graph Neural Networks

Graphs in the field of machine learning represent entities and their relationships, often visually, using nodes (also called vertices) connected by edges. Unlike traditional data structures such as vectors and matrices, graphs excel in modeling complex and non-linear relationships among entities. This makes them particularly suitable for tasks like image and language processing, social network analysis, and drug discovery.

In 1997, Sperduti et al. [167] were pioneers in applying Neural Networks (NNs) to directed acyclic graphs, which motivated early studies on Graph Neural Networks (GNNs). Subsequently, Gori et al. [168] and Scarselli et al. [169] proposed the outline of GNNs. However, these early studies, which relied on recurrent neural networks, were limited by high computational costs.

Inspired by the success of convolutional neural networks in computer vision [69, 170], Kipf et al. introduced Graph Convolutional Networks (GCNs) in 2016 [48]. Encouraged by the application of attention mechanism in natural language processing, Graph Attention neTworks (GATs) has been proposed in 2017 [49, 50]. Then graph generative models emerged [171, 172]. General GNNs follow the Message Passing Neural Network (MPNN) framework [33], with the model's performance influenced by the specifics message-passing algorithm.

Given that molecules can be naturally represented as molecular graphs, with atoms as nodes and bonds as edges, GNNs have demonstrated promising results in various related tasks, including molecular property predictions [33, 34, 162] and molecule graph generation [173, 174]. In this chapter, we present the details in constructing GNNs and their practical applications in downstream tasks.

This chapter is organized as follows: we start by providing an overview of common graph structures, encompassing social networks, citation networks, molecular graphs, and other specialized graphs, in Section 3.1. Following that, in Section 3.2, we introduce various existing techniques in the field of GNNs. In Section 3.3, we introduce our model, Directed Graph Attention Networks (D-GATs), and demonstrate its effectiveness in Section 3.4. Finally, we summarize our findings and present our conclusions in Section 3.5.

3.1 Introduction to Common Graphs

To begin, let us define what a graph is. A graph is a representation of the relationships (edges) among a set of entities (nodes) (refer to Figure 3.1).

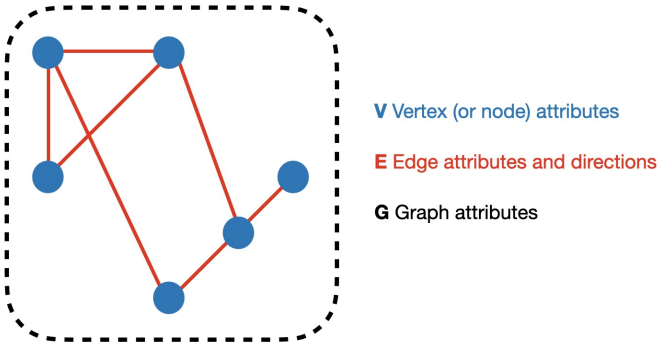

Figure 3.1: **Three common types of attributes in graphs**

To provide a more detailed description of individual nodes, edges, or the entire graph, we can store information within each of these elements. As described in Figure 3.2, we can further customize graphs by introducing directionality to edges, classifying them as either directed or undirected graphs.

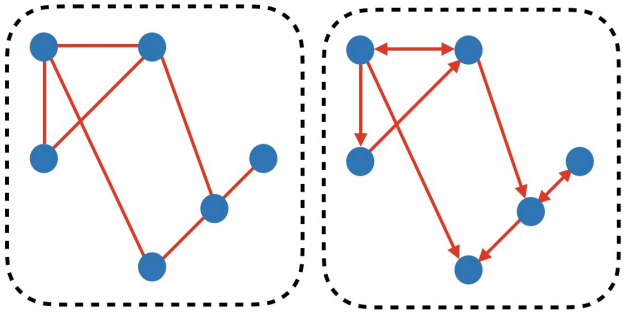

Figure 3.2: **Difference between undirected/directed graphs. (Left)** An example of undirected graph. **(Right)** An example of directed graph where information flux must follow the direction of edges.

Next are three typical graphs: social networks, citation networks or molecular graphs.

3.1.1 Social Network

Social networks (see Figure 3.3) are tools to study patterns in collective behaviour of people, institutions and organizations. We can build a graph representing groups of people by modelling individuals as nodes, and their relationships as edges.

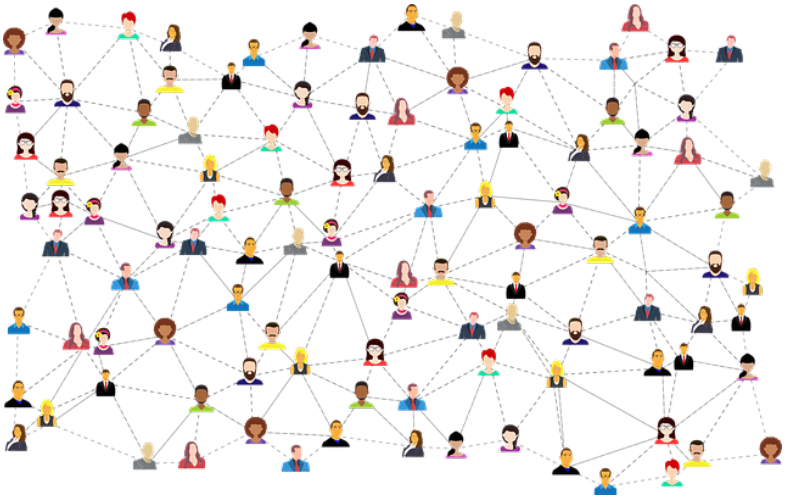

Figure 3.3: **Social Networks.** Image from GDJ, via Pixabay

Social networks can be used to compute individuals' social influence, create highly targeted advertising strategies, and forecast human behavior. A famous example of this is the Zachary's Karate Club, a small-scale social network, wherein a conflict erupts between the club's administrator and instructor [175]. The goal is to predict which side of the conflict each member of the karate club chooses. In Figure 3.4, each node corresponds to a club member, and the connections between nodes illustrate their interactions outside the club.

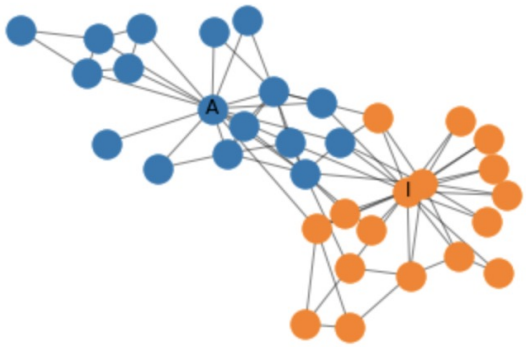

Figure 3.4: **Zachary's Karate Club.** Blue nodes are people that choose administrator's side and yellow nodes are members that follow instructor

3.1.2 Citation Networks

Citation networks, as shown in Figure 3.5, serve as valuable tools for scientists to gather relevant research papers. Scientists routinely cite other scientists' work when publishing academic papers. These networks of citations can be visualized as graphs, where each paper represents a node, and each directed edge signifies a citation from one paper to another. Moreover, we can enhance each node by incorporating additional information about the paper, such as a word embedding of the abstract [176–178] or a categorization into research domains. The citation networks are typically

considered as directed networks due to the directional nature of the citation relationship, which flows from the citing paper to the cited paper.

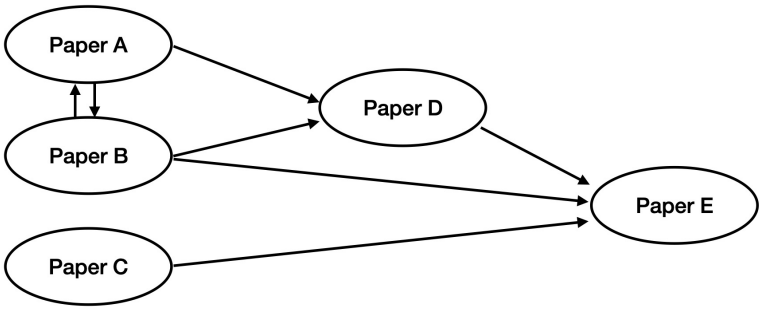

Figure 3.5: **Citation networks.** Paper D cites document A and B. Paper E cites papers B, C and D. Paper A and B cite each other.

3.1.3 Molecular Graphs

Molecules composed of atoms and electrons within three-dimensional (3D) space are the fundamental constituents of matter. Ball-and-stick models, as illustrated in Figure 3.6(a), are a visualization tool for representing the 3D structure of molecules. This model consists of spherical balls representing atoms and sticks denoting the bonds between these atoms. When we abstract away from the spatial arrangement of molecules, it becomes convenient to describe these ball-and-stick models as graphs, as shown in Figure 3.6(b). In this graph representation, atoms serve as nodes, and covalent bonds are represented as edges [33, 179, 180].

While both SMILES and molecular graphs can represent the same molecular structure, each has its distinct advantages and drawbacks. SMILES offers a concise representation that is easily stored and transmitted, but it doesn't capture the molecule's 3D structure. In contrast, molecular graphs can retain geometric information within atom features, but they can be larger and more intricate to manipulate. Additionally, there is ongoing research in the field of text-to-image generation [181–183]. Furthermore, encoding atomic environments into two-dimensional images is also being explored [184].

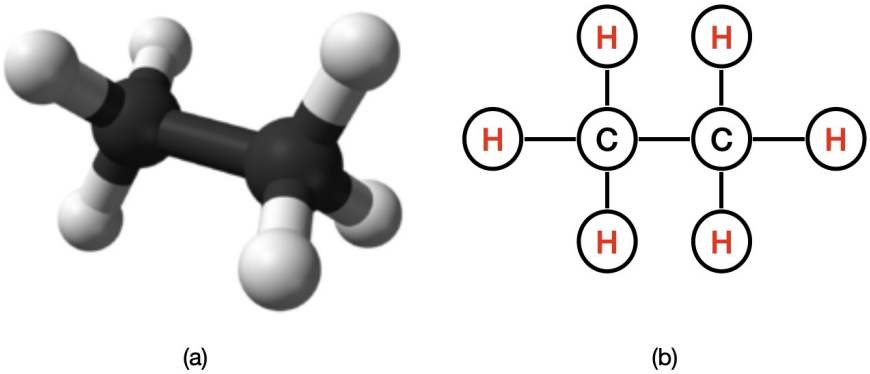

Figure 3.6: **Example of ethane.** **(a)** Ball-and-stick models. **(b)** Graph representation.

Graph representation aligns more closely with human intuition and proves to be more accessible for machine learning models. Consequently, it should be simpler to merge and extract information from molecular graphs. Furthermore, within GNNs, every node (i.e., atom) and each edge (i.e., bond) possess their individual features, simplifying the prediction process for force field parameters. We will study further into this topic in the upcoming chapter.

3.1.4 Other Special Graphs

Indeed, even images and texts can be thought as graphs and processed by GNNs. In the case of images, every pixel can be viewed as a node, interconnected with adjacent pixels via edges, as shown in Figure 3.7. Each non-border pixel maintains 4 or 8 neighboring connections, and the data stored at each node comprises a 3-dimensional vector representing the RGB value of the pixel.

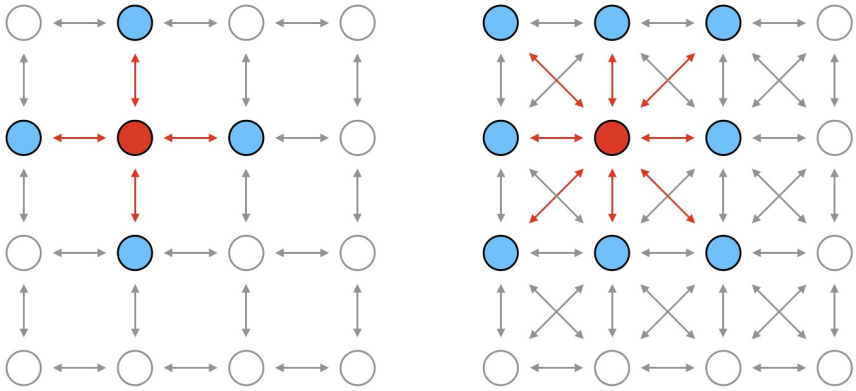

Figure 3.7: **Image to graph.** **(a)** 4-connected pixel adjacency graph **(b)** 8-connected pixel adjacency graph

Text can be thought as a graph in which nodes correspond to vocabularies, and edges are established between adjacent words. Since the order of words is crucial in language, this representation takes the form of a directed graph.

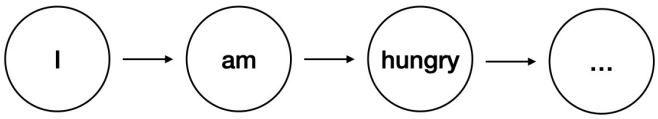

Figure 3.8: **Text to graph**

Certainly, in practical applications, this is not usually how text and images are encoded, as these representations can be redundant for the data with inherent regular structures.

3.2 Different Types of Graph Neural Networks

There are three primary types of prediction tasks associated with graphs: graph-level, node-level, and edge-level tasks. In a graph-level task, the objective is to predict a single property for the

entire graph, such as predicting molecular properties [52, 180]. For node-level tasks, the goal is to predict specific properties for each individual node within a graph, e.g., Zachary's Karate Club task[175]. In edge-level tasks, the aim is to predict properties or the presence of edges within a graph, as seen in link prediction problems [185, 186].

To achieve these goals, various techniques for merging and extracting information are available, including but not limited to Graph Convolutional Networks (GCNs) [46–48], Graph Attention Networks (GATs) [49, 50], and Message Passing Neural Networks (MPNNs) [33]. These technologies play a vital role in addressing a wide range of graph-based challenges.

Here are some notations will be used in this section.

- D_h is the dimension of hidden states.
- N is the number of nodes.
- E is the number of edges.
- n and e in superscript represent atoms and bonds.
- t in superscript indicates the number of layers.
- $\mathcal{N}_i$ denotes the neighbors of node i.
- Graph is noted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges.
- $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix for graph $\mathcal{G}$.
- The degree matrix $D \in \mathbb{R}^{N \times N}$ is a diagonal matrix, defined as $D_{ii} = \sum_j A_{ij}$.
- $H^t = \{h_1^t, h_2^t, ..., h_N^t\}, h_i^t \in \mathbb{R}^{D_h}$ are hidden states in t-th layer.
- $\sigma(\cdot)$ denotes a non-linear activation function.

3.2.1 Graph Convolutional Networks

GCNs define graph convolutions in the spectral domain based on graph Fourier transform. This approach allows spectral-based graph convolutions to be computed by taking the inverse Fourier transform of the product between two Fourier-transformed graph signals. However, GCNs have a limitation in that they do not inherently support edge features, even though bonds play a crucial role in determining molecular properties. Additionally, it is important to note that GCNs can be viewed as a particular type of low-pass filter [187, 188], which means they may omit part of information in graphs.

Generally, if we do not consider the edge features, the multi-layer GCNs follow the layer-wise propagation rule [48]:

$$H^{t+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^tW^t) \tag{3.1}$$

where $H \in \mathbb{R}^{N \times D_h}$ are the node states, I_N is the identity matrix, $\hat{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$ defined as $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ and W^t is a layer-specific trainable weight matrix. We do not expand this part because GCNs are not applied in this thesis.

3.2.2 Graph Attention neTworks

The core concept behind Graph Attention Networks (GATs) revolves around learning a set of attention coefficients that signify the significance of each node's features for a given task. These attention coefficients are acquired through a self-attention mechanism, as discussed in Section 2.2.2. In this mechanism, every node computes a score for all of its neighbors based on their feature representations. These computed scores are then employed to calculate a weighted sum of the features from neighboring nodes. Subsequently, this weighted sum is concatenated with the node's own feature representation. This iterative process is applied to all nodes within the graph, thereby enabling each node to acquire a distinct representation that is tailored to its local neighborhood.

In this section, we present the original GATs, which exclusively process atom features. This distinction is crucial, as it underlies the foundation of our model, D-GATs, as elaborated in Section 3.3.

The core concept behind GATs is to learn a set of attention coefficients that signify the importance of each node's features for a given task. These attention coefficients are learned through a self-attention mechanism, as discussed in Section 2.2.2. In this mechanism, every node computes a score for all of its neighbors based on their feature representations. These scores are then employed to calculate a weighted sum of the features from neighboring nodes. Subsequently, this weighted sum is concatenated with the node's own feature representation. This process is repeated for all nodes in the graph, thereby enabling each node to learn a representation that is specific to its local neighborhoods. Our model (Section 3.3) is based on GATs and here we present the original GATs, which only process the atom features.

To update the node states $h_i^{t+1} \in \mathbb{R}^{D_h}$ for node i in $t+1$ layer, we first calculate the attention coefficients for connected atoms j:

$$e_{ij} = a(W_Q^{t+1} h_i, W_K^{t+1} h_j) \tag{3.2}$$

where function $a(.,.)$ defined as $a : \mathbb{R}^{D_h} \times \mathbb{R}^{D_h} \to \mathbb{R}$ and $W_Q^{t+1}, W_K^{t+1} \in \mathbb{R}^{D_h \times D_h}$ are the trainable parameters.

To make coefficients easily comparable across different nodes, they are normalized across all choices of j using the `SoftMax` function:

$$\alpha_{ij} = \texttt{SoftMax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e_{ik})} \tag{3.3}$$

where $\mathcal{N}(i)$ is the first-order neighbors of node i (not including i).

With $\sigma(\cdot)$ as the activation function and $W_V^{t+1} \in \mathbb{R}^{D_h \times D_h}$, the node states are updated:

$$h_i^{t+1} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W_V^{t+1} h_j^t\Big) \tag{3.4}$$

It is also possible to apply multi-head attention or add more non-linearity to GATs. But we stop the introduction to GATs here.

3.2.3 Message Passing Neural Networks

MPNNs are actually a general framework for supervised and unsupervised learning on graphs [33]. During the message passing phase, hidden states h_i^t at node i in t-th layer are updated based on edge states e_{ij} (between node i and j) and messages m_i^{t+1} according to:

$$m_i^{t+1} = \sum_{j \in \mathcal{N}(i)} M^t(h_i^t, h_j^t, e_{ij}) \tag{3.5}$$

$$h_i^{t+1} = U^t(h_i^t, m_i^{t+1}) \tag{3.6}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i in graph $\mathcal{G}$. M^t is message function and U^t is vertex update function. The initial hidden states are obtained from the inputs: $H^0 = T(X)$ with T is the transform function.

In MPNNs, the edge features can also be updated by introducing hidden states for all edges in the graph $h_{e_{ij}}^t$ and updating them analogously to Equations (3.5) and Equation (3.6) [180, 189].

3.2.4 Directed Message Passing Neural Networks

Functional groups, such as alcohols, ethers, aldehydes, ketones, carboxylic acids, and others, play a crucial role in conferring certain physical and chemical properties to the molecules containing them. Hence, developing an effective algorithm for distinguishing and identifying these sub-structures is essential for advancing molecular representation learning.

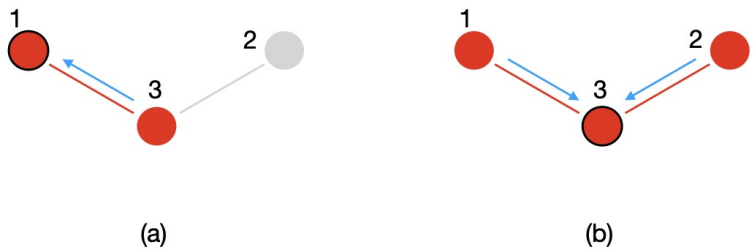

Figure 3.9: **Difference between D-MPNN and MPNN. (a)** In previous GNNs, to update the embedding for node 1, we consider its neighbor, i.e. node 3. Thus the message flows from node 3 to node 1. **(b)** In the layers after sub-figure (a), embedding of node 3 is updated by node 1 node 2. This means the information go through the path: Node 3 $\rightarrow$ Node 1 $\rightarrow$ Node 3 and re-inflow the original node.

Traditional models treat molecular graphs as undirected graphs. However, Directed Message Passing Neural Network (D-MPNN), introduced in works [51, 52], propose directed edges to avoid unnecessary loops during the message-passing phase. Research presented in [190] highlights a key issue known as "over-smoothing", which arises due to excessive mixing of information and noise. The interaction messages transmitted from other nodes can include both helpful information and potentially harmful noise. From this point, directed edges alleviate this over-mixing of information.

The primary distinction between D-MPNN and standard MPNNs lies in the type of messages employed during the message-passing phase. In D-MPNN, messages are associated with directed edges (bonds) rather than nodes (atoms). This design choice is made to prevent "tottering" [74], which can occur in traditional MPNNs, leading to unnecessary loops. As illustrated in Figure 3.9, the message from node 3 $\rightarrow$ 1 will not be disseminated to other nodes in the next iteration in D-MPNN, whereas it will be transmitted back to node 3 in the original MPNNs, generating an unnecessary loop in the message passing trajectory.

During these iterative message-passing excursions, there is a possibility of introducing noise into the graph representation. This edge-based message-passing approach in D-MPNN exhibits similarities to belief propagation in probabilistic graphical models [191]. To learn more about the connection between D-MPNN and belief propagation, please consult [51].

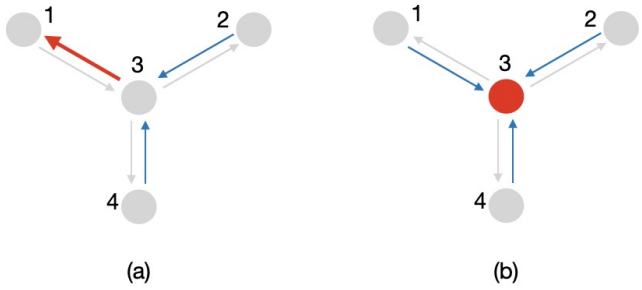

Figure 3.10: **Example of message passing in D-MPNN. (a) Update of edge states:** The edge $3 \rightarrow 1$ is updated by (edge $2 \rightarrow 3$ and edge $4 \rightarrow 3$) **(b) Update of node states:** The node 3 is updated by (edge $1 \rightarrow 3$, edge $2 \rightarrow 3$ and edge $4 \rightarrow 3$)

D-MPNN follows the message passing functions in Equation (3.5) and (3.6). Assuming the inputed node features $F^n = \{F_1^n, F_2^n, ..., F_N^n\}$ and edge features $F^e = \{F_{ij}^e\}$ for all connected nodes i and j, the edge states $\{h_{ij}^{t+1}\}$ at layer $t+1$ are updated by:

$$m_{ij}^{t+1} = \sum_{k \in \mathcal{N}(i) \setminus j} M^t(h_i^t, h_k^t, h_{ki}^t) = \sum_{k \in \mathcal{N}(i) \setminus j} h_{ki}^t \tag{3.7}$$

$$h_{ij}^{t+1} = U^t(h_{ij}^t, m_{ij}^{t+1}) = \sigma(h_{ij}^0 + W_m \cdot m_{ij}^{t+1}) \tag{3.8}$$

where $\sigma(\cdot)$ is the `ReLU` activation function, $W_m \in \mathbb{R}^{D_h \times D_h}$.

The directed edge states are initialized as:

$$h_{ij}^0 = \sigma(W_i \cdot [F_i^n, F_{ij}^e]) \tag{3.9}$$

where $[.,.]$ is the concatenation operation.

The node states $\{h_1, h_2, ..., h_N\}$ are not updated. Instead, they are derived from the initial node features F^n and the edge hidden states at last layer T that direct to the node:

$$h_i = \sigma(W_a \cdot [F_i^n, \sum_{k \in \mathcal{N}(i)} h_{ki}^T]) \tag{3.10}$$

D-MPNN adopts a message-passing paradigm based on updating representations of directed bonds rather than atoms, thereby avoiding unnecessary loops during the message passing phase of the algorithm. it aids in the identification of substructures within molecules. Additionally, the directionality inherent in D-MPNN plays a crucial role in facilitating the charge transfer model, as elaborated in Section 4.2.4.

3.3 Directed Graph Attention neTworks

In this section, we present the details of Directed Graph Attention neTworks (D-GATs), a novel approach that combines the advantages of directed graphs with the efficiency of attention mechanisms, exhibiting a computational complexity of $\mathcal{O}(N)$.

D-MPNN only makes use of simple aggregate functions, which limits model's performance. To enhance performance, we adopt the scaled dot-product attention mechanism [49] for message aggregation. There are two related works that also employ attention mechanisms on directed molecular graphs. Here, we outline the key differences between their models and ours:

- In X. Han's work [54], GEA is based on additive attention mechanism, which is generally less efficient than dot-product attention. In addition, GEA explores various pooling techniques, such as max-pooling, sum-pooling, and set2set [56], for the `ReadOut` function. In contrast, our model, D-GATs, employs a supervirtual node structure, which is more robust.
- In C. Qian's work [53], DGANN employs a similar update function to D-GATs but follows a different logic. DGANN initially updates directed edges, and only the outputs at the last layer are used to update node states and molecule-level representations. In our model, the edge states, node states, and molecular representations are updated in each interaction layer, thus they are tightly coupled.

These distinctions in model design and operation contribute to the unique strengths and capabilities of D-GATs. Our models consist of 3 parts:

Backbone In the networks, chemical bond between two atoms is considered as two different directed bonds. Based on the scaled dot-product attention mechanism, these directed bonds aggregate information from neighboring atoms, which is then used to update the atomic representations. The molecular representation is a virtual atom embedding [192], updated by a `ReadOut` function.

Pre-training To mitigate potential overfitting concerns arising from limited benchmark databases containing only thousands of molecules, we assembled a comprehensive pre-training dataset. This dataset includes all molecules appeared in the experimental sections, along with the ZINC-250K database [133]. For the pre-training tasks, in addition to the masked atom prediction task, we also include molecular properties prediction task for molecules from the ZINC-250K database, to train the supervirtual node.

Fine-tuning For a specific downstream task, we only need fine-tune the whole model or re-train the last layers.

In particular, our work presents the following contributions:

- D-GATs follow the common framework of MPNNs and explore a bond-level message passing algorithm completely relying on scaled dot-product attention mechanism, which outperforms state-of-the-art baselines on 13/15 molecular property prediction tasks (see Table 3.6 and 3.7)on the MoleculeNet benchmark [78].
- Propose a simple but efficient pre-training strategy (see Section 3.4.2).
- The code and pre-trained models of D-GATs are publicly available at `https://github.com/GongCHEN-1995/D-GATs`.

Here are some notations will be used in this subsection.

- D_h is the dimension of hidden states (or called dimension of model).
- t in superscript indicates the number of layers.
- $\mathcal{N}_i$ denotes the neighbors of node i.
- $\{h^t_{\vec{p}(ij)}\}$ for connected atoms i and atom j. $h^t_{\vec{p}(ij)} \in \mathbb{R}^{D_h}$ are bond states in t-th layer.
- $\{h^t_1, h^t_2, ..., h^t_N\}, h^t_i \in \mathbb{R}^{D_h}, i = 1, ..., N$ are atom states in t-th layer.
- $\mathcal{S} \in \mathbb{R}^{D_h}$ are molecular representations in t-th layer.
- $\sigma(\cdot)$ denotes a non-linear activation function.
- $n/e/\mathcal{S}$ in superscript represent atoms/bonds/molecular representations correspondingly.

The notations in Equation (3.7-3.10) are directly from paper [52] and they are different from those in D-GATs. This is because we use e_{ij} to indicates undirected bond while $\vec{p}(ij)$ indicates directed bond. Additionally, we use attention mechanism to replace summation as aggregate function, making our functions more complicated. Therefore, we have applied some modifications to notations.

It should be noted that all trainable parameters in D-GATs are initialized by Kaiming initialization [130].

3.3.1 Initialization of Input Features

We use RDKit [75] to process SMILES and extracting atom/bond features. These features and molecular graph (in Lewis structure [76]) are the inputs to D-GATs. Table 3.1 lists the required input features to D-GATs. Since categorical data contains label values that cannot be directly processed by our model, we employ one-hot encoding to convert categorical data to numerical data.

Atom Features	Size(127)	Descriptions
atom symbol	100	[From H to Fm] (one-hot)
degree	6	number of covalent bonds [0, 1, 2, 3, 4, 5] (one-hot)
formal charge	1	electrical charge (integer)
radical electrons	1	number of radical electrons (integer)
hybridization	8	[unspecified, s, sp, sp2, sp3, sp3d, sp3d2, other] (one-hot)
chirality	4	[unspecified, tetrahedral_CW, tetrahedral_CCW, other] (one-hot)
number of hydrogen atoms	5	[0, 1, 2, 3, 4] (one-hot)
ring	1	whether the atom is in ring [0/1] (one-hot)
aromaticity	1	whether the atom is part of an aromatic system [0/1] (one-hot)
Bond Features	**Size(12)**	**Descriptions**
bond type	4	[single, double, triple, aromatic] (one-hot)
conjugation	1	whether the bond is conjugated [0/1] (one-hot)
ring	1	whether the bond is in ring [0/1] (one-hot)
stereo type	6	[StereoNone, StereoAny, StereoZ, StereoE, Stereocis, Stereotrans] (one-hot)

Table 3.1: Inputed atomic and bond features to graphs.

Given the input atom features $F^n = \{F^n_1, F^n_2, ..., F^n_N\}, F^n_i \in \mathbb{R}^{127}, i = 1, ..., N$ and the input bond features $F^e = \{F^e_1, F^e_2, ..., F^e_E\}, F^e_p \in \mathbb{R}^{12}, p = 1, ..., E$, where

- n and e in superscript represent atoms and bonds.

- N is the number of atoms in molecule and 127 is the number of possible atom features.
- E is the number of bonds and 12 is the number of possible bond features. We write $p = p(i, j)$ to indicate the bond p that links atoms i and atom j. Note that $p(i, j) = p(j, i)$.

Initialization of Directed Bonds States: we construct the initial directed bond states from atom i to atom j as:

$$h^0_{\vec{p}(ij)} = W^e_T([F^n_i, F^e_{p(i,j)}, F^n_j]) \tag{3.11}$$

where $[.,.]$ is the concatenation operation and $W^e_T \in \mathbb{R}^{D_h \times 266}$ is a learnable matrix to convert the concatenation of $F^n_i, F^e_{p(i,j)}$ and F^n_j into a vector in dimension D_h. D_h is the dimension of model and in our model $D_h = 512$. Even though $F^e_{p(i,j)}$ does not contain any directionality, the two inputed atom features F^n_i and F^n_j cannot be commuted and thus introduce directionality by indicating the start atom and the end atom correspondingly.Note that $h^0_{\vec{p}(ij)} \neq h^0_{\vec{p}(ji)}$.

Initialization of Atom States: the initial atom states $h^0 = \{h^0_i | i = 1, 2, ..., N\}$ are transformed from atom features F^n:

$$h^0_i = W^n_T F^n_i \tag{3.12}$$

where $W^n_T \in \mathbb{R}^{D_h \times 127}$ is a learnable matrix to convert the atom features into a vector in dimension D_h

Initialization of Molecular Representations: we introduce a molecular feature, following the notion of supervirtual node $\mathcal{S}$ introduced in Attentive FP [166] that connects all atoms of the molecule. The initialized molecular representation $\mathcal{S}^0 \in \mathbb{R}^{D_h}$ is a trainable vector used to represent molecule and will be updated with attention mechanism.

3.3.2 Update of Representations

In this subsection, we will talk about how to update the states through scaled dot-product attention mechanism. The update follows the order showed in Figure 3.11(b). In each interaction layer, we apply three times attention mechanism to update directed bond states, atom states and molecular representations separately. The trainable parameters in layer $t + 1$ for attention mechanism are:

$$W^{t+1}_{Q^e}, W^{t+1}_{K^e}, W^{t+1}_{V^e}, W^{t+1}_{Q^n}, W^{t+1}_{K^n}, W^{t+1}_{V^n}, W^{t+1}_{Q^{\mathcal{S}}}, W^{t+1}_{K^{\mathcal{S}}}, W^{t+1}_{V^{\mathcal{S}}} \in \mathbb{R}^{D_h \times D_h}$$

The trainable parameters in multilayer perception (MLP) are:

$$W^e_1, W^e_2, W^n_1, W^n_2, W^{\mathcal{S}}_1, W^{\mathcal{S}}_2 \in \mathbb{R}^{D_h \times D_h}$$

$\sigma(\cdot)$ is the Rectified Linear Unit (`ReLU`) activation function.

Update of Directed Bond States

Note $\mathcal{E}_{ij} = \{\vec{p}(ij)\} \cup \{\vec{p}(ki) | k \in \mathcal{N}(i), k \neq j\}$ where $\mathcal{N}(i)$ denotes the neighbor atoms of atom i. Following the framework and notations in [33, 52], we compute the bond messages $m^{t+1}_{\vec{p}(ij)}$ by the equations:

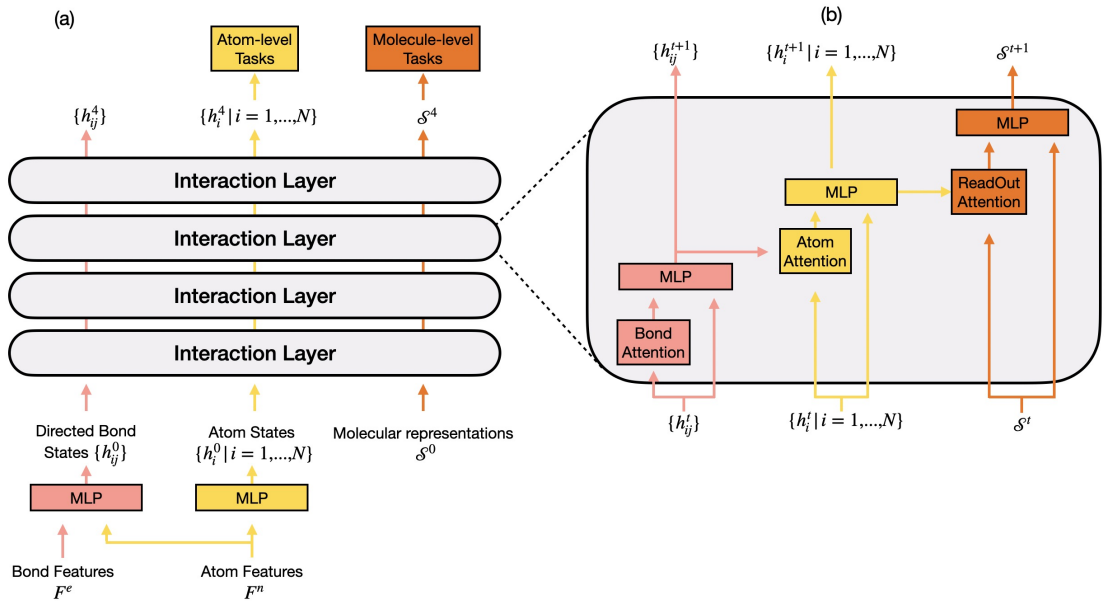

Figure 3.11: **Framework of D-GATs. (a)** The inputs, 4 interaction layers and outputs. **(b)** Details in each interaction layer

$$m_{\vec{p}(ij)}^{t+1} = M_e^{t+1}(h_q^t | q \in \mathcal{E}_{ij}) = \sum_{q \in \mathcal{E}_{ij}} \alpha_{\vec{p}(ij),q}^{t+1}(h_q^t W_{V^e}^{t+1}) \tag{3.13}$$

The attention-based message functions M_e^{t+1} involves the coefficients $\alpha_{\vec{p}(ij),q}^{t+1}$ $(q \in \mathcal{E}_{ij})$ by:

$$\alpha_{\vec{p}(ij),q}^{t+1} = \texttt{Softmax}_q(e_{\vec{p}(ij),z}^{t+1} | z \in \mathcal{E}_{ij}) = \frac{\exp(e_{\vec{p}(ij),q}^{t+1})}{\sum_{z \in \mathcal{E}_{ij}} \exp(e_{\vec{p}(ij),z}^{t+1})} \tag{3.14}$$

$$e_{\vec{p}(ij),q}^{t+1} = \frac{(h_{\vec{p}(ij)}^t W_{Q^e}^{t+1})(h_q^t W_{K^e}^{t+1})^T}{\sqrt{D_h}} \tag{3.15}$$

Next is a MLP where the messages are used to update directed bond states by update functions U_e^{t+1}:

$$h_{\vec{p}(ij)}^{t+1} = U_e^{t+1}(h_{\vec{p}(ij)}^t, m_{\vec{p}(ij)}^{t+1}) = W_2^e(\sigma(W_1^e(\texttt{LayerNorm}(h_{\vec{p}(ij)}^t + m_{\vec{p}(ij)}^{t+1})))) \tag{3.16}$$

And `LayerNorm`, a type of normalization technique, is from [77].

Compared to undirected graphs, directed graphs prevent the information from being repeatedly passed back to its source and thus reduce noise. Besides, unless the atom information going through a ring structure, the same substructures always result in the same bond states. For instance, in Figure 3.12 (a) and (b), h_{67}^t in two molecules are the same if $t \leq 7$. When $t \geq 8$, i.e., with 8 layers, $h_{\vec{p}(67)}^t$ are different in (a) and (b) because the influence of atom 8 arrives at $h_{\vec{p}(67)}^t$ through the chain $8 \to 7 \to 6 \to 5 \to 3 \to 1 \to 2 \to 4 \to 6$ after t=8 steps.

Additionally, the computational cost for directed graphs is quadrupled because the number of bonds is doubled (one undirected bond generates two directed bonds).

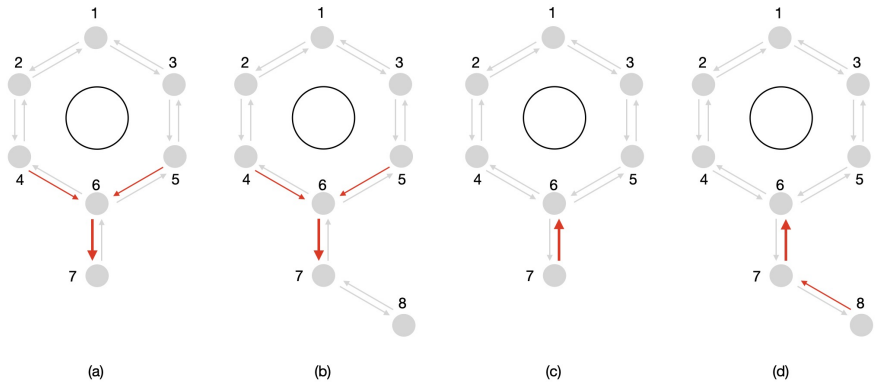

Figure 3.12: **Example of directed message flow.** **(a)** and **(b)**: $h^{t+1}_{\vec{p}(67)}$ is updated by $[h^t_{\vec{p}(67)}, h^t_{\vec{p}(46)}, h^t_{\vec{p}(56)}]$, thus they have the same embeddings for $t \leq 7$. **(c)** and **(d)**: $h^t_{\vec{p}(76)}$ are different for $t > 0$ due to the existence of $h^t_{\vec{p}(87)}$

Update of Atom States

Followed by the update of directed bond states, atom messages m_i^{t+1} are updated through vertex message functions M_n^{t+1}:

$$m_i^{t+1} = M_n^{t+1}(h_i^t, h^{t+1}_{\vec{p}(ji)} | j \in \mathcal{N}(i)) = \alpha^{t+1}_{i,i}(h_i^t W^{t+1}_{V^n}) + \sum_{j \in \mathcal{N}(i)} \alpha^{t+1}_{i,j}(h^{t+1}_{\vec{p}(ji)} W^{t+1}_{V^n}) \tag{3.17}$$

For $j \in \mathcal{N}(i) \cup \{i\}$, the attention weights are computed as:

$$\alpha^{t+1}_{i,j} = \texttt{Softmax}_j(e^{t+1}_{i,k} | k \in \mathcal{N}(i) \cup \{i\}) = \frac{\exp(e^{t+1}_{i,j})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e^{t+1}_{i,k})} \tag{3.18}$$

$$e^{t+1}_{i,k} = \begin{cases} \frac{(h_i^t W^{t+1}_{Q^n})(h_i^t W^{t+1}_{K^n})^T}{\sqrt{D_h}} & \text{k=i} \\ \frac{(h_i^t W^{t+1}_{Q^n})(h^{t+1}_{\vec{p}(ki)} W^{t+1}_{K^n})^T}{\sqrt{D_h}} & \text{k} \neq \text{i} \end{cases} \tag{3.19}$$

Next is to update atom states in MLP:

$$h_i^{t+1} = U_n^{t+1}(h_i^t, m_i^{t+1}) = W_2^n(\sigma(W_1^n(\texttt{LayerNorm}(h_i^t + m_i^{t+1})))) \tag{3.20}$$

As presented in Figure 3.10(b), during the update process, atom states collect the information flows in and are independent to the information flows out. The atom states are used to update molecular representation $\mathcal{S}^{t+1}$. Moreover, as the atom states merge atoms' chemical environment, they can also be applied to do atom-level tasks (e.g. to classify atom type) or to recover masked atoms in pre-training stage.

Update of Molecular Representations

There exists a virtual node connected to all atoms in molecule (see Figure 3.13) and it is used as molecule-level representation. Known the updated atom states h_i^{t+1}, the molecular representations $\mathcal{S}^{t+1}$ are updated by `ReadOut` function defined as:

$$m^{t+1} = \texttt{ReadOut}^{t+1}(\mathcal{S}^t, h_i^{t+1} | i = 1, 2, ..., N) = \alpha_{\mathcal{S}}^{t+1}(\mathcal{S}^t W_{V\mathcal{S}}^{t+1}) + \sum_{j=1}^{N} \alpha_i^{t+1}(h_i^{t+1} W_{V\mathcal{S}}^{t+1}) \quad (3.21)$$

for $i \in [1, N] \cup \{\mathcal{S}\}$:

$$\alpha_i^{t+1} = \texttt{Softmax}_i(e_k^{t+1} | k \in [1, N] \cup \{\mathcal{S}\}) = \frac{\exp(e_i^{t+1})}{\sum_{k \in [1,N] \cup \{\mathcal{S}\}} \exp(e_k^{t+1})} \quad (3.22)$$

$$e_i^{t+1} = \begin{cases} \frac{(\mathcal{S}^t W_{Q\mathcal{S}}^{t+1})(\mathcal{S}^t W_{K\mathcal{S}}^{t+1})^T}{\sqrt{D_h}} & \text{k=}\mathcal{S} \\ \frac{(\mathcal{S}^t W_{Q\mathcal{S}}^{t+1})(h_k^{t+1} W_{K\mathcal{S}}^{t+1})^T}{\sqrt{D_h}} & \text{k} \in \text{[1,N]} \end{cases} \quad (3.23)$$

Finally, the molecular representations are:

$$\mathcal{S}^{t+1} = U_{\mathcal{S}}^{t+1}(\mathcal{S}^t, m^{t+1}) = W_2^{\mathcal{S}}(\sigma(W_1^{\mathcal{S}}(\texttt{LayerNorm}(\mathcal{S}^t + m^{t+1})))) \quad (3.24)$$

The supervirtual node offers higher expressive power compared to the summation or averaging of atom states. The final molecular representations are the learned graph-level vectors that encode structural information about the molecular graph and chemical information including the functional groups, followed by a task-dependent feed-forward neural network for prediction.

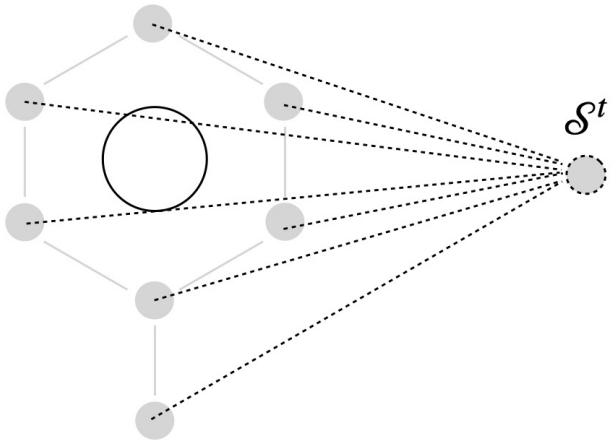

Figure 3.13: **Supervirtual node $\mathcal{S}^t$.** It is connected to all atoms to update the molecular representations

3.4 Experiments

In this section, we present the details of our experiments, including the databases to test, the strategy to pre-train, fine-tune our model and their performance on a variety of benchmarks.

3.4.1 Databases and Metrics

Here we use the same databases presented in Section 2.3.1, including quantum mechanics tasks, physical chemistry tasks, biophysics tasks, physiology tasks (see Table 3.2) and pre-training database ZINC-250K. All datasets still do the scaffold splitting [150] to increase the challenge for learning algorithms. The ratio of training, validation and test sets is 8:1:1.

Classification Tasks					
Physiology			Biophysics		
Dataset	Number of Molecules	Number of Tasks	Dataset	Number of Molecules	Number of Tasks
BBBP	2039	1	BACE	1513	1
SIDER	1427	27	HIV	41127	1
Tox21	7831	12	MUV	93087	17
ToxCast	8575	2	PCBA	437929	128
ClinTox	1478	617			
Regression Tasks					
Physical Chemistry			Quantum Mechanics		
Dataset	Number of Molecules	Number of Tasks	Dataset	Number of Molecules	Number of Tasks
ESOL	1128	1	QM7	6830	1
FreeSolv	642	1	QM8	21786	12
Lipo	4200	1	QM9	133885	12
ZINC-250K	249455	3			

Table 3.2: Databases used for downstream tasks for D-GATs.

We take the same metrics explained in section 2.3.2, i.e., mean absolute error (MAE) or root-mean-square error (RMSE) for regression problems and area under the receiver operating characteristic curve (ROC-AUC) for classification problems.

3.4.2 Pre-Training

Since the majority of the 15 databases used for testing contain only thousands of molecules, there is a high risk of overfitting, which can lead to a decline in model performance on test set. To mitigate this issue, we employ pre-training and fine-tuning strategy, which offers several benefits such as improved generalization, faster convergence, and better understanding of molecular structures. Pre-training is a form of transfer learning. It involves initially training a model on a larger or related database or task, followed by fine-tuning on smaller, more specific databases or tasks. The pre-training step allows the model to learn general features that can be transferred to the downstream tasks, which can help extract high-level features from raw molecular graphs and reduce the reliance on extensive training data.

In the context of molecular properties prediction, pre-training a model could involve designing the self-supervised (or called unsupervised) tasks that requires the model to predict some aspect of the molecule, such as the presence of certain substructures or simply, the atom features or bond features. This pre-training task would provide the GNNs with additional knowledge about molecular structures that it can leverage when fine-tuning on a specific downstream task, such as predicting the solubility or toxicity of a given molecule.

Our molecular pre-training dataset encompasses all public databases utilized in model validation, with the exception of the PCBA database, from which we randomly selected 40,000 molecules due to its substantial size. Additionally, we incorporate the ZINC-250K database [Irwin and Shoichet, 2012]. To manage computational resources effectively, we filter out overly complex molecules (those with more than 60 heavy atoms, excluding hydrogen) and overly simple ones (those with fewer than 10 heavy atoms).

The molecular pre-training dataset is based on all public databases utilized in model validation, with the exception of the PCBA database, from which we randomly selected 40,000 molecules due to its substantial size. Additionally, we incorporate the ZINC-250K database [132]. To manage computational resources effectively, we filter out overly complex molecules (those with more than 60 heavy atoms, excluding hydrogen) and overly simple ones (those with fewer than 10 heavy atoms).

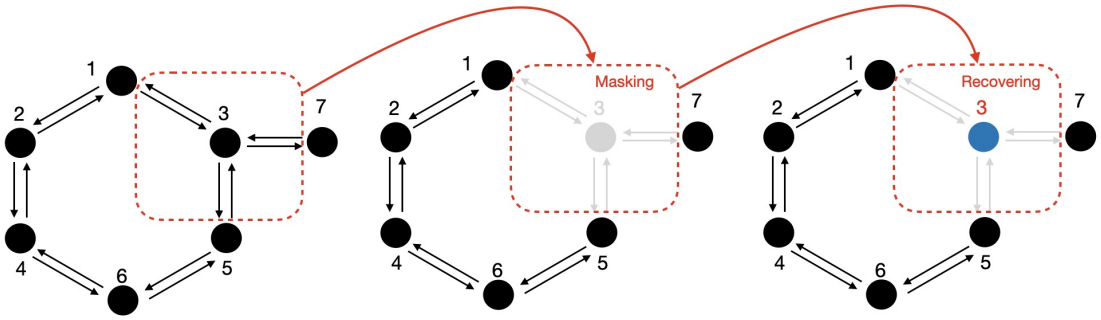

Figure 3.14: **Pre-training tasks for D-GATs. Masking:** The features of chosen atoms and connected bonds are masked (in grey). **Recovering:** Only recover the masked atom's features (in blue).

While NLP and GNNs operate on distinct types of data inputs, they require different pre-training strategies. In the previous chapter, we introduced three pre-training tasks for NLP models: recovering masked tokens, connection classification, and same molecule classification. However, for D-GATs, the connection is indicated directly by edges. We do not consider the atom coordinates, thus translation and rotation do not affect the input features. Every molecule has its unique molecular graph inputs and same molecule classification cannot be applied. Consequently, the sole unsupervised task remaining is the recovery of masked nodes and edges.

Following the set in BERT [164], 20% of the input atoms are chosen for possible replacement. Among these, 80% are marked as [MASK] element and have no other atom features, 10% remain unchanged an the remaining 10% are replaced by randomly generated atom features. Additionally, all bonds connected to these 20% of atoms are marked as masked bonds, which remove all bond features (see Figure 3.14).

Number of layers	4	Dimension of model	512	Initial learning rate	5E-04	Training data	3.6E5
Number of heads	8	Dimension of MLP	512	Smallest learning rate	1E-06	Validation data	4.5E4
Number of atom features	127	Dropout	0.1	Gamma	0.999	Test data	4.5E4
Number of bond features	12	Minimum heavy atoms	10	Maximum heavy atoms	60	Batch size	Flexible

Table 3.3: Hyper-parameters of pre-training model for D-GATs

While we have masked certain atoms and edges, our primary aim is to recover atomic features exclusively. According to our message passing algorithm presented in Section 3.3, atom

representations are updated by bond representations but independent to the update of bond representations. Therefore, successfully recovering atom features is based on the ability to correctly recover the masked bond features and there is no more need to recover bond features.

The unsupervised task of recovering atom features operates at the atom level and does not directly influence the training of the `ReadOut` function. As illustrated in Figure 3.11(a), atom-level tasks enable the parameter training for directed bond states and atom states, while not involving the `ReadOut` function. Consequently, graph-level tasks are introduced to pre-train parameters for molecular representations.

To accomplish this, we employ a molecular properties prediction task, utilizing only the ZINC-250K database (as shown in Figure 3.15). The task involves predicting key values, including LogP (logarithm of solubility), SAS (synthetic accessibility score), and QED (quantitative estimate of drug-likeness). To ensure data integrity and prevent leakage, we strictly avoid using molecular properties from the test dataset for model pre-training.

For the tasks to recover atom features, we use Binary Cross Entropy loss (Equation (2.19)) to determine whether the atom is in ring or in aromatic environment. To predict the element type, degree of atoms, hybridization, chirality, and the number of connected hydrogen atoms, we use cross entropy loss (Equation (2.20)). In the case of tasks related to recovering formal charge and radical electrons for masked atoms, as well as regression tasks for predicting molecular properties within the ZINC-250K dataset, we utilize Mean Square Error (Equation (1.15)) as the loss function.

Pre-training model is composed of the stacked D-GATs (for extracting bond, atom, and molecule features) and feed-forward NNs (for converting representations from D-GATs into atom features or molecular properties). For various downstream tasks shown in Table 3.2, with parameters in pre-trained D-GATs being slightly optimized, only a single layer feed-forward NNs for fine-tuning tasks need to be trained to transform molecular representations into molecular properties. Our D-GATs incorporate four interaction layers, employ a model dimension (D_h) of 512, a dropout rate of 0.1, and utilize 8 heads for the multi-head attention mechanism (N_h).

Number of Atoms	$N \leq 15$	$15 < N \leq 30$	$30 < N \leq 45$	$45 < N \leq 60$	$N > 60$
Batch Size	1024	512	350	256	32

Table 3.4: Batch size of pre-training data for D-GATs

The model is pre-trained for 100 epoch using 1 NVIDIA V100 GPU of 16GB. Pre-training took approximately 2 days. Initial learning rate for all parameters is $1E-4$, that decays by gamma (see Table3.3) every 200 steps. The decay will be stopped after learning rate is less than smallest learning rate. Additionally, we used the Adam optimiser [111] with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both pre-training and fine-tuning on all tasks. To reduce overfitting, set `DropOut` $= 0.1$.

	Element	Aromatic	Ring	Degree	Hybridization	Chirality	H
Accuracy	92.4%	100%	100%	99.5%	99.4%	98.2%	93.7%

Table 3.5: Accuracy of pre-training model for D-GATs

Based on the pre-training results in Table 3.5, it is evident that our model achieves notably high accuracy when predicting masked atom features. Degree (99.5%), Hybridization (99.4%), and Chirality (98.2%) all demonstrate remarkable accuracy, indicating the model's ability to capture and learn the underlying patterns and relationships between atoms. Notably, our pre-training models achieve perfect accuracy in predicting Aromatic and Ring features, showcasing

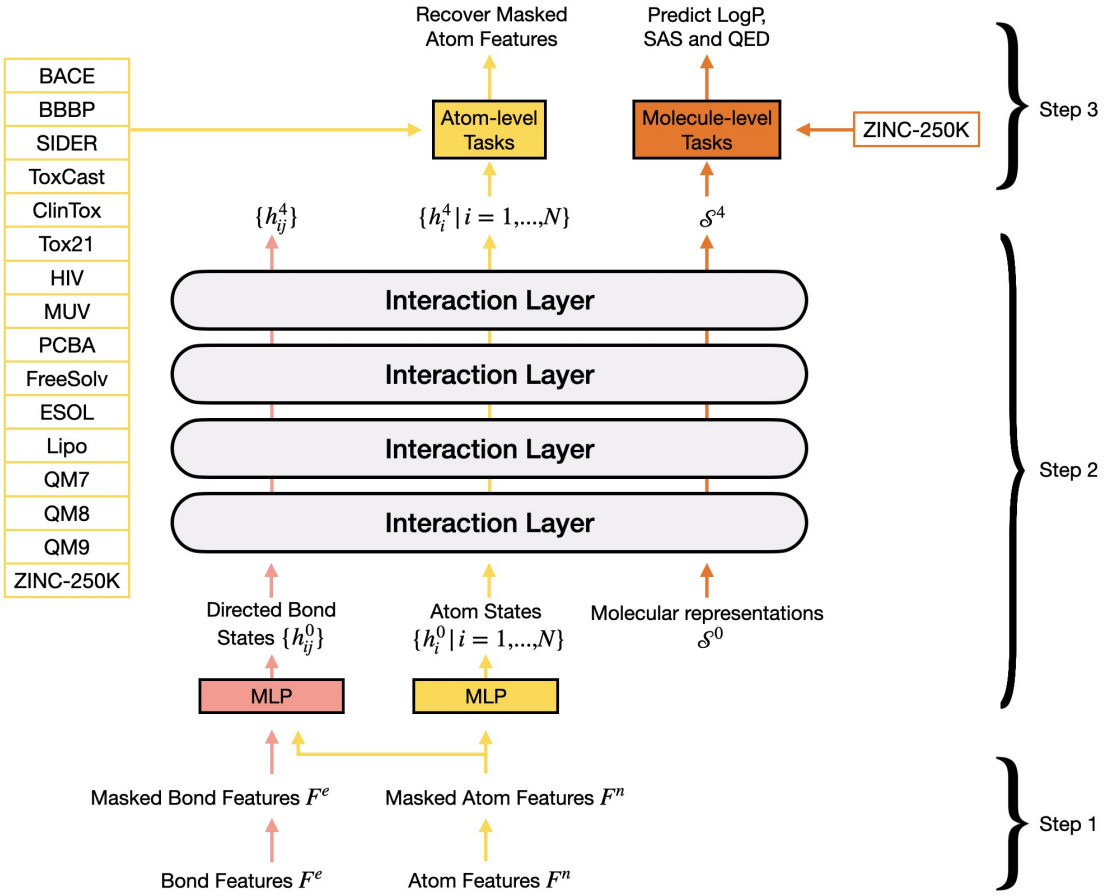

Figure 3.15: **Pre-training stage for D-GATs. Step1:** Part of the inputed atom features and bond features are masked. **Step2:** The masked features are passed to interaction layers and give the final bond states, atom states and molecular representations. **Step 3:** The molecules in left 16 databases are used to recover masked features and the molecular properties prediction task is only related to database ZINC-250K

the model's proficiency in recognizing distinctive patterns associated with aromatic and ring molecules.

However, the prediction of element and the number of connected hydrogen atoms exhibits comparatively lower accuracy, falling below 94% simultaneously. This can be attributed to the close relationship between hydrogen count and atom type. When an atom is masked, the possible and reasonable substitutions are not unique. This may explain the relatively lower accuracy compared to other features such as aromaticity and ring characteristics, which are more straightforward and less dependent on the surrounding atoms. Therefore, it is necessary to consider the underlying complexity of the atomic interactions and the potential limitations of the pre-training task when interpreting the results.

In summary, these pre-training outcomes underscore D-GATs' capacity to effectively capture intricate atomic patterns and relationships, which proves valuable for downstream tasks, including molecular property prediction.

3.4.3 Experimental Results

We have trained one pre-training model. As suggested by the MoleculeNet [78], for each database, we generate three downstream linear classifiers with random seeds. The mean and standard deviation of the results are reported in Table 3.6 and Table 3.7. The best results are marked in bold.

Classification Tasks ROC-AUC % (higher is better) in form Average (Standard Deviatioin)

Dataset	**BBBP**	**SIDER**	**Tox21**	**ToxCast**	**ClinTox**	**BACE**	**HIV**	**MUV**	**PCBA**	**Avg**
Number of molecules	**2,039**	**1,427**	**7,831**	**8,575**	**1,478**	**1,513**	**41,127**	**93,087**	**437,929**	
Number of prediction tasks	**1**	**27**	**12**	**617**	**2**	**1**	**1**	**17**	**128**	
GROVER_{large}	67.8(0.2)	62.2(1.9)	73.5(0.1)	65.3(0.1)	78.8(0.7)	81.4(1.3)	72.8(0.5)	67.8(1.3)	83.1(0.5)	72.5
NLP model (Functional Group)	64.7(1.4)	62.8(0.5)	73.2(0.3)	63.9(0.4)	82.4(0.5)	82.9(1.1)	74.5(0.5)	73.0(0.7)	83.9(0.2)	73.5
NLP model (Element)	68.1(0.6)	60.1(0.6)	74.4(0.1)	62.6(0.3)	84.5(0.2)	74.8(0.3)	76.5(0.2)	76.8(0.4)	84.1(0.1)	73.5
AttentiveFP	65.2(0.9)	60.7(1.6)	76.7(0.5)	67.4(0.1)	82.8(0.6)	80.9(0.3)	75.4(0.9)	73.0(0.5)	80.3(0.6)	73.6
D-MPNN	71.2(0.3)	60.2(0.4)	75.1(0.2)	64.3(0.4)	89.6(0.2)	80.0(0.2)	76.4(1.4)	75.3(1.8)	86.2(0.3)	75.4
Pretrain-GCN	70.5(0.9)	62.5(0.2)	75.8(0.3)	65.4(0.1)	63.5(1.8)	84.1(0.4)	76.9(0.7)	79.6(0.2)	84.7(0.1)	73.7
Pretrain-GIN	70.4(0.3)	62.9(0.1)	78.1(0.5)	65.7(0.5)	73.1(1.2)	84.4(0.2)	79.6(0.1)	82.0(0.1)	**86.5(0.1)**	75.9
GEM	71.6(1.3)	60.6(1.0)	77.4(0.7)	67.5(0.5)	89.3(0.2)	82.8(1.2)	78.0(0.8)	74.7(0.7)	86.3(0.4)	76.5
D-GATs	**71.7(0.2)**	**65.8(0.6)**	**78.6(0.2)**	**67.7(0.2)**	**90.9(0.7)**	**84.5(0.3)**	**79.8(0.1)**	**82.5(0.6)**	85.6(0.1)	**78.6**

Table 3.6: Results of molecular property classification tasks for D-GATs

We compare D-GATs with multiple baselines, including our NLP models presented in Section 2.3.4, supervised and pre-training baselines. D-MPNN [52] and AttentiveFP [166] are supervised GNNs methods. GROVER [34], PretrainGNN [162] and GEM [35] are pre-training methods.

Regression Tasks (lower is better) in form Average (Standard Deviation)

	RMSE			MAE		
Dataset	**ESOL**	**FreeSolv**	**Lipo**	**QM7**	**QM8**	**QM9**
Number of molecules	**1,128**	**642**	**4,200**	**6,830**	**21,786**	**133,885**
Number of prediction tasks	**1**	**1**	**1**	**1**	**12**	**12**
GROVER_{large}	0.907(0.002)	2.888(0.014)	0.817(0.015)	92.4(6.7)	0.0226(0.0017)	5.836(0.111)
NLP model (Functional Group)	1.023(0.028)	2.803(0.042)	0.775(0.011)	84.5(1.8)	0.0203(0.0004)	4.314(0.087)
NLP model (Element)	1.204(0.031)	3.282(0.103)	0.821(0.009)	104.9(1.6)	0.0214(0.0002)	4.431(0.128)
AttentiveFP	0.915(0.022)	1.945(0.094)	0.729(0.011)	**69.2(2.54)**	0.0181(0.0001)	4.166(0.146)
D-MPNN	1.075(0.005)	2.140(0.070)	0.688(0.009)	97.6(1.5)	0.0179(0.0004)	6.240(0.287)
Pretrain-GCN	1.255(0.014)	2.095(0.114)	0.770(0.005)	83.8(2.2)	0.0200(0.0001)	8.006(0.066)
Pretrain-GIN	1.150(0.023)	2.763(0.075)	0.759(0.012)	94.1(3.8)	0.0201(0.0005)	8.450(0.112)
GEM	0.835(0.025)	1.899(0.054)	0.680(0.009)	77.8(2.4)	0.0174(0.0001)	3.894(0.056)
D-GATs	**0.743(0.017)**	**1.653(0.072)**	**0.676(0.008)**	87.1(3.0)	**0.0172(0.0001)**	**3.056(0.142)**

Table 3.7: Results of molecular property regression tasks for D-GATs

Our results suggest the following trends:

- D-GATs demonstrate superior performance compared to our NLP models. Across 15 downstream databases, D-GATs outperform baseline models in 13 instances. Notably, D-GATs exhibit significant improvements, particularly in databases like ClinTox and FreeSolv. In classification databases (as seen in Table 3.6), D-GATs give the most promising performance, leading to an increase in average ROC-AUC of 2.1% over the previous SOTA results.

- For D-GATs, a straightforward pre-training strategy involving the recovery of masked atom inputs and supervised learning for the `ReadOut` component suffices to uncover intrinsic rules within molecules. This pre-training stage effectively mitigates overfitting and accelerates fine-tuning in downstream tasks. Remarkably, the pre-training model generalizes well to larger molecules with more atoms than those encountered during pre-training, as exemplified by benchmarks like SIDER, where the maximum number of heavy atoms can reach 495.

- The incorporation of attention mechanism in D-GATs surpasses the performance of D-MPNN [52].
- Despite its successes, D-GATs fail to beat state-of-the-art results in the QM7 databases (see Table 3.7) due to overfitting challenges. Additionally, in the case of the PCBA database, the presence of imbalanced samples and unlabelled data significantly harms model performance.

3.5 Conclusions

In this chapter, we explore the flexibility of graphs as they are capable of representing a wide range of data. By breaking down data subjects into nodes and edges, we can effectively describe various types of data. We introduce common graph types, including social networks, citation networks, molecule graphs, and other specialized graphs.

To merge and extra information in graphs, several typical techniques are proposed: Graph Convolutional Networks (GCNs), Message Passing Neural Networks (MPNNs), Graph Attention neTworks (GATs) and etc. In molecular properties prediction tasks, these graph-based models outperform Natural Language Processing (NLP) models due to the rich structural information present in molecular graphs. While SMILES and molecular graphs theoretically possess the same expressive power, graphs are more straightforward and interpretable for machine learning models. Furthermore, in the upcoming chapter, assigning atom types aligns well with Graph Neural Networks (GNNs), as it is an atom-level task.

Traditional GNNs treat molecular graphs as undirected graphs, but directed bonds, as proposed in D-MPNN [52], mitigate unnecessary loops during message passing. In this thesis, we choose D-GATs over other GNNs because of the flexibility of the attention mechanism. With an attention-based aggregator, message flow within directed bonds enables highly efficient parallelization. Moreover, the dynamic highlighting and weighting of salient information, in a similar manner as it does in the human brain, make attention an attractive concept in machine learning. Our results (as shown in Table 3.6 and 3.7) demonstrate that D-GATs outperform D-MPNN due to the attention mechanism. Our model adheres to the common MPNN framework and avoids complex operations, maintaining a manageable size of approximately 100 MB.

Our evaluation spans 15 diverse benchmarks, encompassing molecular properties across various domains, from thousands to hundreds of thousands of compounds. Our extensive analysis demonstrates the superiority of the message passing algorithm in D-GATs for learning molecular representations compared to other GNNs. Remarkably, D-GATs achieve this with basic atom and bond features, surpassing strong baseline models in both classification and regression tasks. D-GATs consist of three vital components: an attention-based scheme for updating bond and atom representations, a `ReadOut` function for extracting molecular representations, and a linear classifier for downstream tasks. Our results underscore the potential of D-GATs as a formidable tool for molecular property prediction.

It is important to note that D-GATs are specifically designed for small-sized graphs commonly found in molecular properties. While slightly less computationally efficient than undirected graph models, the manageable extra cost is acceptable. However, the presence of rings in graphs can affect directed message flow, necessitating careful consideration of model depth. Consequently, D-GATs are best suited for molecular graphs and may not be optimal for large or dense graphs like social networks.

Typically, atom type based force field parameterization engines [193] such as those used in AMBER [194] or GAFF [81] assign parameters based on templates (for biopolymer residues or solvents) or through chemical perception algorithms. The atom type is decided by the atom environment thus it is considered as an atom-level task. D-GATs output the corresponding atom

states for each atom, allowing for the derivation of atom types or force field parameters without additional operations. Further details will be presented in the next chapter.

An important future direction of our work is to enhance the generalization ability of D-GATs through improved pre-training strategies. While our thesis follows a masking-based pre-training strategy inspired by BERT [164], advanced and intricate pre-training strategies exist, such as Context Prediction [162] which allows the model to match the chemical environment, or the geometry-enhanced learning strategy proposed in [35], which leverages 3D information such as bond lengths and angles hold promise. Additionally, exploring more sophisticated message passing algorithms, such as higher-order messages [93], can further enhance model expressiveness without adding more layers.

Chapter 4

Force Field Parameterization by Machine Learning

In the previous chapters, we have successfully applied Machine Learning (ML) techniques to analyze molecules and predict their properties. Nevertheless, our focus was predominantly centered on molecular-level attributes, treating each molecule as an independent entity. In reality, molecules are composed of atoms, which constitute the fundamental building blocks of all matter in the physical world and they cannot be simply studied as isolated entities. Consequently, it is imperative to study molecules from both microscopic and macroscopic perspectives in order to gain a deeper understanding of the underlying physics and chemistry. To bridge this gap between the microscopic and macroscopic worlds, molecular dynamics simulations serve as a crucial tool.

However, the efficiency and accuracy of molecular dynamics simulations depend on a multitude of factors.These factors encompass the complexity of system, the scale of simulation, the specific properties being investigated, and the calculations methods.

Density Functional Theory (DFT) stands as a first-principles method grounded in quantum mechanics, offering remarkable precision. Nevertheless, its computational demands can be substantial. On the other hand, Force Fields (FFs) rely on empirical potentials and simplified functional forms to describe interatomic interactions. FFs serve as the most efficient computational methods, albeit at the expense of reduced accuracy. Polarizable FFs aim to mitigate certain limitations of conventional FFs by incorporating polarization effects. Meanwhile, ML Potentials can yield results comparable to DFT for numerous properties while remaining computationally more tractable. However, their reliability hinges on the quality and size of training data.

This chapter introduces our Graph-Based Force Fields (GB-FFs) model, designed to reparameterize existing non-polarizable FFs (refer to Figure 4.1 for an efficiency comparison) with the goal of achieving higher accuracy.

This chapter is organized as follows: we begin by providing an overview of the background of molecular mechanics and an analysis of the advantages and disadvantages of existing methods in Section 4.1. Following that, in Section 4.2, we present the architecture of our GB-FFs model, which is divided into three components: the molecule processing model, the symmetry-preserving parameter generator, and the charge transfer model. In Section 4.3, we introduce our training strategies and showcase the performance of our models across various benchmarks. Lastly, we summarize our findings and present our conclusions in Section 4.4. Supplementary information is provided in Section 4.5.

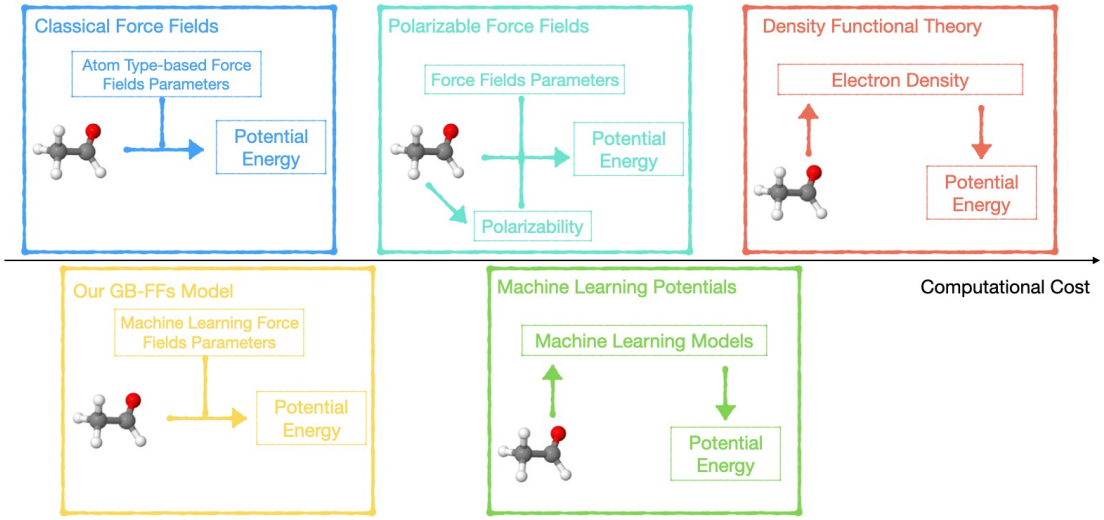

Figure 4.1: **Comparison of computational efficiency:** Classical Force Fields, our Graph-Based Force Fields models, Polarizable Force Fields, Machine Learning Potentials and Density Functional Theory.

4.1 Introduction to the Background of Molecular Mechanics

In computational chemistry, two distinct approaches, Quantum Mechanics (QM) and Molecular Mechanics (MM), are employed to model and simulate molecular interactions. QM methods are typically used for small-scale systems, providing precise and intricate details regarding electronic structures and molecular attributes. These encompass properties such as bond dissociation energies, reaction barriers, spectroscopic properties, and more [195]. However, the computational cost associated with ab-initio methods within QM restricts their application to relatively diminutive systems, typically composed of fewer than a thousand atoms, thereby imposing limitations on the study of larger molecular systems.

As an alternative, MM approaches have emerged as valuable tools. They rely on physically-motivated functional representations to model potential energy surfaces [79]. These MM methods are meticulously parameterized to align with ab-initio energies or to reproduce experimental data. This parameterization offers a computationally cheaper alternative for simulating diverse systems, ranging from biological entities to polymers and complex materials.

4.1.1 Molecular Mechanics and Force Fields

MM stands as a computational method employed to investigate the behavior of molecules and materials at the atomic and molecular scale. Rooted in classical mechanics and founded on the Born-Oppenheimer approximation [1], MM finds widespread utility across diverse systems, ranging from simple molecules like hydrocarbons to intricate biomolecular complexes and composite material structures.

[1]The Born-Oppenheimer approximation, introduced by Max Born and J. Robert Oppenheimer in 1927 [196], exploits the significant mass disparity between nuclei and electrons. This allows the nuclei's coordinates to be considered as parameters, separating from the electron dynamics. This approximation forms the basis for FFs models, where functional expressions are parameterized with respect to nuclei coordinates.

The origins of MM trace back to the early 20th century, when scientists initiated the use of theoretical models to explore the behavior of atoms and molecules. The contemporary MM approach extends beyond basic atom and bond representations in classical chemistry. Remarkably, the first successful models of proteins and DNA were crafted in the 1950s, relying on hand-made primitive atom and bond models constructed from wood and wire. Nobel Prize authors Pauling et al. [197] and Watson and Crick [198] were among the pioneers in this area. Over the past half-century, the adoption of MM has surged significantly, resulting in a manifold increase in publications employing this methodology.

Force fields (FFs), as elucidated by Poltev et al. [80], serve as the cornerstone of MM. Under the context of MM, the energy of a molecular system is described as the sum of interatomic potentials. These potentials are typically expressed through mathematical equations and are characterized by empirical parameters, which are known as FF parameters. These parameters are painstakingly determined through a fusion of experimental observations and quantum mechanical calculations. Consequently, FFs calculate the energies and forces between atoms according to their nuclear coordinates. Typically, FFs incorporate the following features:

- Individual atoms are represented as discrete particles.
- Each particle is assigned a specific radius (Van der Waals radius), alongside polarizability and a constant net charge. These values are either derived from quantum calculations or experimental investigation.
- Interactions arising from chemical bonds are depicted as springs, with an equilibrium distance set to match either the experimentally observed or theoretically calculated bond lengths.

The development of powerful computers from 1980s allows for the use of MM simulations to study larger and more complex molecular systems. For different purposes and compounds, there are different FFs. Here are some famous examples:

AMBER [1–3] Assisted Model Building and Energy Refinement is developed by Peter Kollman's group. The term "Amber" refers to a set of FFs for the simulation of biomolecules as well as a package of molecular simulation programs which includes source code and demos. This project began in the late 1970's and now is the most popular MM simulation program for wide class of biomolecules including proteins and nucleic acids. The current AMBER versions are strongly aimed at simulations (molecular dynamics, evaluations of free energy changes) of biomolecules in water solutions. Both explicit solvent models (using TIP3P [199], TIP4P [199], TIP5P [200], and SPC [201]water models as well as models of some organic solvents) and implicit ones are supported.

CHARMM [202–205] Chemistry at HARvard Molecular Mechanics is developed by Martin Karplus's group. It refers to the FFs and a molecular simulation program with broad application to many-particle systems with a comprehensive set of energy functions, a variety of enhanced sampling methods, and support for multi-scale techniques including QM/MM, MM/CG, and a range of implicit solvent models. The authors of CHARMM elaborate their own polarizable models of water based on classical Drude oscillator.

ECEPP [206, 207] Empirical Conformational Energy Program for Peptides is developed by F.A. Momany, H.A. Scheraga and colleagues, specifically for the modeling of peptides and proteins. It uses fixed geometries of amino acid residues to simplify the potential energy surface. Thus, the energy minimization is conducted in the space of protein torsion angles.

MMFF94 [208, 209] Merck Molecular Force Field is developed by Merck Research Laboratories. The core portion of MMFF94 has primarily been derived from high-quality computational quantum chemistry data (up to MP4SDQ/TZP level of theory) for a wide variety of chemical systems of interest to organic and medical chemists.

GROMOS [210–212] GROningen MOlecular Simulation is developed by van Gunsteren et al. It is a force field that comes as part of the GROMOS software, a general-purpose molecular dynamics computer simulation package for the study of biomolecular systems. The GROMOS force fields are not pure all-atom force fields because aliphatic CH_n groups are treated as united atoms.

OPLS [213, 214] Optimized Potential for Liquid Simulations is developed by William L. Jorgensen. The OPLS-AA (all atom) FFs retain most of bond stretch and angle bending parameters from AMBER all-atom FFs, but torsion and nonbonding constants are reparametrized utilizing both experimental and *ab-initio* (RHF/6-31G* level) data.

AMOEBA [215–219] Atomic Multipole Optimized Energetics for Biomolecular Applications is developed by Pengyu Ren and Jay W. Ponder. AMOEBA includes full intramolecular flexibility, permanent atomic monopole, dipole, and quadrupole moments placed on each atomic center and buffered 7-14 potential for pairwise atom-atom van der Waals interactions. Polarization effects are explicitly treated in the AMOEBA force field via mutual induction of dipoles at atomic centers. It was parametrized by hand to fit results from *ab-initio* calculations on gas phase clusters up to hexamers [215] and temperature and pressure dependent bulk phase properties [216].

4.1.2 Polarizable Force Fields

Classical FFs primarily rely on stretching, bending, and dihedral terms to characterize intramolecular interactions, complemented by fixed-charge Coulomb potentials and Lennard-Jones interactions to model intermolecular interactions. Renowned for their computational efficiency, classical FFs enable simulations of large systems over extended time scales. However, their simplistic functional forms lack the capacity to account for vital polarization and many-body effects, which are essential for precisely elucidating intricate phenomena like pi-stacking or allosteric effects. Additionally, due to their empirical nature, classical FFs suffer from inaccuracies and inconsistencies.

In recent years, propelled by advances in computational capabilities, there has been a growing interest in developing polarizable FFs. Noteworthy examples include AMOEBA [217–219], CHARMM Drude [220], and SIBFA [221, 222]. These polarizable FFs are proficient in capturing polarization effects, particularly critical in systems dominated by electrostatic interactions, such as protein-ligand binding or aqueous solutions. They have been developed to explicitly incorporate polarization and many-body effects. This augmented flexibility and accuracy, however, do entail a higher computational cost relative to classical FFs. Nevertheless, polarizable FFs provide a more comprehensive representation of intermolecular interactions and prove particularly well-suited for investigating intricate systems.

In polarizable FFs, atomic charges are not held constant, but instead, dynamically adjust in response to the local electric field. This dynamic adjustment enables a more precise description of electrostatic interactions between molecules, in contrast to non-polarizable FFs, where atomic charges remain fixed. Furthermore, polarizable FFs possess the capability to account for charge transfer phenomena occurring between atoms and molecules, a level of detail unattainable in non-polarizable FFs.

Nevertheless, the development of polarizable FFs presents a formidable challenge, demanding an accurate modeling of the polarization behavior. Achieving this often necessitates a larger

number of parameters and incurs an increased computational burden, which can limit the practicality of polarizable FFs in simulations involving larger and more complex systems. Our work does not employ polarizable FFs, thus we won't delve into further details on this topic.

4.1.3 Machine Learning Potentials

For an extended period, computer simulations in the fields of chemistry, molecular biology, and materials science heavily relied on computationally intensive electronic structure calculations, such as Density-Functional Theory (DFT), or less accurate empirical potentials, often derived from physical approximations and intuition. However, with the development of modern machine learning (ML) techniques, a paradigm shift has occurred. ML potentials have emerged as a bridge between the accuracy of DFT and the efficiency of empirical potentials [21–25]. These ML potentials leverage flexible functional forms from the ML domain, such as deep neural networks, graph networks, or kernel models, to accurately fit *ab-initio* energies or forces. They describe atomic interactions based on atomic positions and nuclear charges, enabling their use in large-scale simulations like molecular dynamics [223]. Importantly, ML potentials operate many orders of magnitude faster than DFT calculations, with minimal loss of accuracy. Furthermore, ML potentials excel at capturing intricate interactions, including polarization effects [224, 225] and challenging metal-ligand interactions [24, 226, 227], which are often problematic for traditional FFs.

A profound comprehension of atomic-level systems, spanning from small molecules to bulk materials, necessitates an understanding of the Potential Energy Surface (PES). The PES contains all the critical information, encompassing stable and metastable structures, atomic forces governing dynamics at finite temperatures, transition states, barriers governing reactions and structural transitions, and atomic vibrations. Most contemporary ML potentials can be effectively applied to systems comprising a vast number of atoms, often reaching tens of thousands. Importantly, they maintain the requisite translational, rotational, and permutational invariances of the PES.

Since the seminal work by Doren and colleagues in 1995 [228], numerous methods have emerged in the field of ML potentials for molecular simulations. These methods include Neural Network Potentials (NNP) [229–232], Gaussian Approximation Potentials (GAP) [233], kernel-based approaches such as Gradient Domain Machine Learning (GDML) [30, 234], Spectral Neighbor Analysis Potentials (SNAP) [235, 236], Moment Tensor Potentials (MTP) [237], Atomic Cluster Expansion (ACE) [238], Graph Networks [239], Kernel Ridge Regression (KRR) methods [145], Atomic Permutationally Invariant Polynomials (aPIP) [240], and Support Vector Machines (SVM) [241]. Typically, ML potentials exhibit the following characteristics:

They propose analytical expressions for the Potential Energy Surface (PES). Their analytical derivatives provide insights into the forces acting on the constituent atoms. These potentials refrain from imposing any assumptions about the functional form, apart from the implicit approximations inherent in the chosen reference electronic structure method.

- They propose analytical expressions for the PES.
- Their analytical derivatives provide insights into the forces acting on the constituent atoms.
- These potentials avoid imposing any assumptions about the functional form, apart from the implicit approximations inherent in the chosen reference electronic structure method.

ML potentials offer several advantages, including their highly adaptable structure that allows for precise representation of reference data and their broad applicability, making them suitable for

modeling a wide range of bonding and atomic interactions, from covalent and metallic bonding to dispersion forces. Moreover, ML potentials boast a well-defined analytical form, enabling consistent computation of properties such as energies, forces, and the stress tensor.

Nonetheless, the accuracy of ML potentials depends on several critical factors, including the quality of training data as well as the architecture of ML model. These dependencies give rise to limitations in terms of transferability and interpretability. Furthermore, the models used in ML potentials often exhibit high complexity, rendering them challenging to interpret. This complexity can obscure the underlying physics and chemistry of the systems under investigation. ML potentials also tend to be orders of magnitude slower than widely used FFs, even when running on powerful hardware accelerators. This reduced speed arises from the need to assess the chemical environment for each energy or force calculation. In contrast, FF parameters are assigned just once for each system and can subsequently be used to compute energies, forces, or conduct molecular simulations. Thanks to optimizations in standard molecular mechanics packages, FFs operate with exceptional efficiency.

Additionally, it is essential to note that molecular energies and atomic forces are fundamental molecular properties. Consequently, our model D-GATs (discussed in Section 3.3) also have the potential to learn ML potentials. The primary challenge lies in incorporating geometric information effectively.

4.1.4 Tinker-HP

Tinker [4, 5] is a software package designed for polarizable molecular dynamics simulations and to polarizable MM, widely used in academic research and industrial applications. It is developed by Jay W. Ponder and his colleagues at Washington University in St. Louis, and is freely available for academic use.

In Tinker, user can perform energy minimization and molecular dynamics computations on full or partial structures, over Cartesian, internal or rigid body coordinates, and including a variety of boundary conditions and crystal cell types. It has a number of options for parallel computing, allowing for faster simulations of large systems. There are also programs to generate timing data and to verify potential function derivatives for coding errors.

Tinker has a user-friendly interface to simplify the customizations and offers a range of analysis tools for visualizing and interpreting simulation results. Users only need to write basic "front-end" programs and the source code is possible to be modified according to their wishes. These make Tinker be adapted to specific needs and preferences. The present version of the package has been successfully transferred to a wide variety of computers with minimal modifications.

Tinker is known for its flexibility in handling a wide range of molecular systems, including proteins, nucleic acids, carbohydrates, and small organic molecules. It offers a range of FFs options, including Amber and CHARMM potentials, MM2, MM3, OPLS, MMFF, Liam Dang's polarizable potentials. Besides, it is possible to create custom FFs. Tinker is also known for its own advanced AMOEBA, AMOEBA+, and HIPPO (Hydrogen-like Intermolecular Polarizable Potential) FFs.

Tinker-HP [242, 243] is an interdisciplinary project led by Jean-Philip Piquemal in collaboration with Jay W. Ponder and Pengyu Y. Ren. It is an evolution of the popular Tinker package that conserves it simplicity of use but brings new capabilities allowing performing very long molecular dynamics simulations on modern supercomputers that use thousands of cores and multiple GPUs. Tinker-HP proposes a high performance scalable computing environment for polarizable and classical FFs, giving access to large systems up to millions of atoms. It can be used on supercomputers as well as on lab clusters.

Due to the high efficiency of Tinker-HP, our code will be performed on this platform.

4.2 Force Field Parameterization by Machine Learning

Parameterizing a FF is a challenging task, as the accuracy and transferability of the FF heavily rely on the quality of its parameters. This process is often time-consuming, taking several years, and relies on empirical heuristics, experimental data, and computational data. Classical FFs have established robust parameterization procedures, such as Antechamber for GAFF [92]. However, parameterizing polarizable FFs, like AMOEBA, presents greater challenges. While automatic parameterization procedures like poltype [244] and its recent extension, poltype2 [245], have been successfully applied to various compounds, including small organic molecules, proteins, and nucleic acids, the process remains less straightforward for polarizable FFs.

Furthermore, these FFs rely on local frames, known as atom types or atom classes, to assign parameters (e.g., bonds, angles). To enhance the generalization and reliability of FFs, there is a trend to expand the atom type space. However, this leads to a proliferation of possible valence compositions, introducing significant complexity into the parameter fitting process. Even with modern parameter optimization frameworks [88] and sufficient data, FF parameters defined by fixed atom types can sometimes exhibit limited transferability.

Traditionally, FFs have been fitted to experimental data and continue to be so. But the advancements in computational capacity and improved scaling of *ab-initio* methods have provided new opportunities. Consequently, there has been an increasing effort to leverage ML for predicting FF parameters while preserving the predefined functional form of the potential. Wang et al. [28] pioneered the combination of graph neural networks (GNNs) and automatic differentiation to predict FF parameters, focusing on intramolecular interactions and demonstrating the effectiveness of GNNs in predicting FF parameters based on potential energies.

Building upon these advancements, we introduce the Graph-Based Force Fields (GB-FFs) model, a universal framework for FF parameterization. The GB-FFs model will be made publicly available at `https://github.com/GongCHEN-1995/GB-FF-Model` [2]. GB-FFs automatically derives accurate FF parameters using only basic atom features and bond features. It offers a continuous alternative to traditional discrete atom typing schemes, eliminating the need to assign atom types and obtaining FF parameters directly from atomic representations. This approach extends the generalization of FFs. Additionally, we propose new functions to enrich the FF functional forms.

4.2.1 General AMBER Force Field (GAFF)

Our model is built upon existing FFs, and in this subsection, we introduce the FFs to be reparameterized. Over the years, AMBER has undergone continuous refinement and improvement. Alongside its FFs, the AMBER software package offers a wide range of tools for conducting molecular dynamics simulations, including tasks such as energy minimization, equilibration, production simulations, and the analysis of simulation results. The AMBER package enjoys widespread adoption within the biomolecular simulation community and has played a important role in facilitating numerous groundbreaking discoveries in the fields of biochemistry and biophysics.

One of the most prominent classical FFs for simulating organic molecules is the General AMBER Force Field (GAFF), which was developed in 2004 by Junmei Wang and Peter A. Kollman as an extension of the Amber force field. GAFF is designed to be compatible with

[2] The code is expected to be released in November 2023.

existing versions for proteins and nucleic acids and includes parameters for a broad spectrum of organic molecules comprising elements C, N, O, H, S, P, F, Cl, Br, and I [81].

Our primary objective is to optimize the parameterization process of GAFF, with the goal of enhancing simulation performance by providing well-suited parameters. It is important to note that these optimizations exclusively pertain to parameter adjustments, leaving the fundamental simulation logic and workflow of GAFF unchanged. This means that the existing AMBER package can seamlessly utilize our optimized parameters without any modifications. This accessibility greatly broadens the reach and impact of our work, as users can readily employ these optimized parameters in their simulations using the familiar AMBER package, streamlining their research processes.

GAFF encompasses an extensive set of parameters for bond stretching, angle bending, dihedral angles, and non-bonded interactions. These parameters enable precise modeling and simulation of organic molecule behavior across diverse conditions, including high-pressure and low-temperature scenarios. Thanks to its computational efficiency, relative reliability, and especially its straightforward functional forms, GAFF has gained widespread adoption in numerous popular molecular simulation software packages, including AMBER [2], GROMACS [246], CHARMM [205], and Tinker-HP [25, 242, 243].

Notably, one of GAFF's major strengths lies in the public availability of its parameters, making it accessible to the scientific community. This accessibility further promotes its widespread utilization within the scientific community.

$$E_{GAFF} = E_{stretching} + E_{bending} + E_{dihedrals} + E_{non-bonded} \tag{4.1}$$

with

$$\begin{aligned} E_{stretching} &= \sum_{bonds} K_r(r - r_{eq})^2 \\ E_{bending} &= \sum_{angles} K_\theta(\theta - \theta_{eq})^2 \\ E_{dihedrals} &= \sum_{n=1}^{4} \frac{V_n}{2}\left[1 + \cos(n\phi - \gamma)\right] + \frac{V_2}{2}\left[1 + \cos(2\varphi - \pi)\right] \\ E_{non-bonded} &= \sum_{i<j}\left[\epsilon_{ij}\left(\frac{\sigma_{ij}^{12}}{R_{ij}^{12}} - 2\frac{\sigma_{ij}^{6}}{R_{ij}^{6}}\right) + \frac{q_i q_j}{\varepsilon R_{ij}}\right] \end{aligned} \tag{4.2}$$

As shown in Equation (4.1) and (4.2), in GAFF, the interactions for bond stretching and angle bending are modeled using harmonic potentials, resulting in non-reactive behavior and simplifying the parameterization process significantly. The torsional potential is expressed as a Fourier series. For non-bonded interactions, Van der Waals (VdW) interactions are described by the 12-6 Lennard-Jones potential [82, 83]. Electrostatic potential, on the other hand, is governed by Coulomb's law.

- r is the bond length. θ is the bond angle. ϕ is the torsional angle. φ is the improper torsional angle. R_{ij} is the distance between non-bonded atoms i and j. These values are determined by the molecular conformations. Known the coordinate of each atom, the molecular dynamics package computes these values at high efficiency.
- r_{eq} and θ_{eq} are equilibrium structural parameters. K_r, K_θ, V_n are force constants. n is multiplicity and γ is phase angle for torsional angle parameters. Specifically, γ is set to 0 for odd values of n and π for even values.

- Dihedral terms contain the torsional terms (the first terms in $E_{dihedrals}$) and improper torsional terms (the second terms in in $E_{dihedrals}$). To distinguish between them, we note torsional angle as ϕ [3] and note improper torsional angle as φ [4]. Since torsional energy is angle-dependent, it is expressed as a four-term Fourier series.

- The VdW parameters ϵ, σ, and charge q characterize the non-bonded potentials. Partial charges are assigned using a restrained electrostatic potential fit (RESP) model [84, 85]. GAFF consists of 97 atom types, resulting in a total of 4753 possible non-bonded interaction pairs. To avoid the impracticality of assigning unique VdW parameters to each interaction pair, GAFF assigns VdW parameters for each atom type and applies Lorentz-Berthelot rules [86, 87] presented in Equations (4.3) and (4.4) to provide a reasonable approximation.

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \tag{4.3}$$

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{4.4}$$

The parameterization process involved over 3,000 MP2/6-31G* optimizations and 1,260 MP4/6-311G(d,p) single-point calculations. The parameterization of GAFF begins with the assignment of partial charges. In its early stages, Hartree-Fock (HF) calculations with the 6-31G* basis set were employed to generate electrostatic potentials, from which RESP charge fits were derived. This process, although accurate, proved to be computationally expensive, particularly for large molecules or when dealing with a substantial number of molecules. Consequently, it led to the development of the AM1-BCC charge scheme. This scheme approximates HF/6-31G* RESP calculations by first determining charges using the AM1 semi-empirical model and then refining them with bond charge corrections [247, 248].

GAFF's equilibrium bond lengths, denoted as θ_{eq}, are determined through a combination of experimental data obtained from X-ray, neutron diffraction experiments and MP2/6-31G* computations. On the other hand, bond angle parameterization relies on references from the Cambridge Structure Database, empirical rules, and MP2/6-31G* calculations. The strategy for developing torsional angle parameters involves conducting torsional angle scans and fitting the parameters to accurately replicate the rotational profiles obtained from MP2/6-31G* calculations. The VdW parameters used in GAFF are consistent with those employed by AMBER.

Since 2015, the second generation of GAFF (GAFF2) has been publicly available through the AMBER program and AmberTools. Both the first and second generations of GAFF share the same functional forms. However, the primary distinction lies in the second generation's expanded set of atom types and the adjustment of specific parameters. For the purposes of this paper, our focus is exclusively on the second generation of GAFF. Once the GB-FFs model framework is validated, it has the potential to be extended to other non-polarizable FFs, while generalization to polarizable FFs is still under development.

The GAFF parameters, denoted as $\{K_r, r_{eq}, K_\theta, \theta_{eq}, V_n, \epsilon, \sigma\}$, are directly extracted from "gaff2.dat" file (version: GAFF2.11) according to corresponding atom types. The conventional FFs typically rely on legacy atom typing schemes for parameter assignment, following these steps:

- Classifying atoms into discrete atom types based on their chemical environments.

[3] Considering four atoms connected sequentially in the order A-B-C-D (such as H1-C1-C2-O in Figure 4.3), torsional angle ϕ is the angle between two planes defined as A-B-C and B-C-D.

[4] Improper term is for sp^2 center atom C and his connected atoms A,B,D (Usually, we note improper atoms as A-B-C-D and the third atom C is the center atom. For example, the H1-H2-C1-H3 in Figure 4.3 is a improper torsion term.). φ is defined as the angle between the line C-D and the plane A-B-C.

- Determining the types of interaction pairs by composing atom types.
- Assigning parameters related to atoms, bonds, angles, and dihedrals according to the FF's parameter table.

To streamline this process, an automated, table-driven procedure called "antechamber", which is a part of the AMBER program, has been developed. Antechamber can efficiently assign atom types, charges, and FF parameters to nearly any organic molecule [92]. If we have installed AmberTools and have the "A.pdb" file, we can use the following command to generate GAFF parameters file "A.ac". Further details can be found at `https://ambermd.org/antechamber/ac.html`.

```
antechamber -i A.pdb -fi pdb -o A.ac -fo ac -c bcc -pf y -at gaff2
```

4.2.2 Molecules Processing Model

Atom Features	Size(38)	Descriptions
atom symbol	11	[UNK],[H],[C],[N],[O],[F],[P],[S],[Cl],[Br],[I] (one-hot)
degree	6	number of covalent bonds [0, 1, 2, 3, 4, 5] (one-hot)
hybridization	8	[unspecified, s, sp, sp2, sp3, sp3d, sp3d2, other] (one-hot)
chirality	4	[unspecified, tetrahedral_CW, tetrahedral_CCW, other] (one-hot)
ring	1	whether the atom is in ring [0/1] (one-hot)
aromaticity	1	whether the atom is part of an aromatic system [0/1] (one-hot)
formal charge	7	[-3,-2,-1,0,1,2,3] (one-hot)
Bond Features	**Size(12)**	**Descriptions**
bond type	4	[single, double, triple, aromatic] (one-hot)
conjugation	1	whether the bond is conjugated [0/1] (one-hot)
ring	1	whether the bond is in ring [0/1] (one-hot)
stereo type	6	[StereoNone, StereoAny, StereoZ, StereoE, Stereocis, Stereotrans] (one-hot)

Table 4.1: Input features to GB-FFs model.

Graph neural networks (GNNs) have proven to be efficient and powerful tools for detecting chemical environments and extracting molecular properties [34, 35, 73, 162]. Additionally, GNNs have demonstrated their potential in capturing atomic and bond representations [34, 162, 249], making them valuable for improving FFs.

Assigning atom types to atoms and deriving FF parameters are typical atom-level tasks. Inspired by the concept of directed bonds proposed in D-MPNNs [52], our model adopts Directed Graph Attention Networks (D-GATs) as the backbone architecture (as presented in Section 3.3). D-GATs excel in detecting local chemical environments and eliminate unnecessary message flow compared to other ML-based molecular processing models. They have outperformed state-of-the-art baselines on 13 out of 15 molecular property prediction tasks. We use RDKit [75] to process SMILES and extracting atom/bond features. These features, along with the molecular graph in Lewis structure [76], form the model inputs.

To enhance robustness, we employ the Smooth Maximum Unit (SMU) [89] as the activation function. SMU smoothly approximates various functions, including the Maxout [90] family, ReLU, Leaky ReLU, and their variants. This choice is motivated by the need for a smooth activation function, as we aim to predict a set of parameters that enable molecular dynamics simulations to closely approximate *ab-initio* data. The molecular PES is highly sensitive to these predicted parameters, and our experiments have shown that the discontinuity in the Maxout function can hurt the convergence of model's loss.

To be compatible with GAFF, we consider compounds composed of elements C, N, O, H, S, P, F, Cl, Br, and I. RDKit extracts fundamental atomic and bond features (refer to Table 4.1), which are then fed into our Graph-Based Force Fields (GB-FFs) model. The model's outputs consist of atomic representations and bond representations. Unlike the original D-GATs, which are designed to predict molecular properties and include a ReadOut function to generate molecule-level representations, in this chapter, we focus solely on obtaining directed bond representations to capture chemical information and atomic representations to predict FF parameters (as illustrated in Figure 4.2(c)). As a result, the ReadOut function is omitted from the Molecule Processing model.

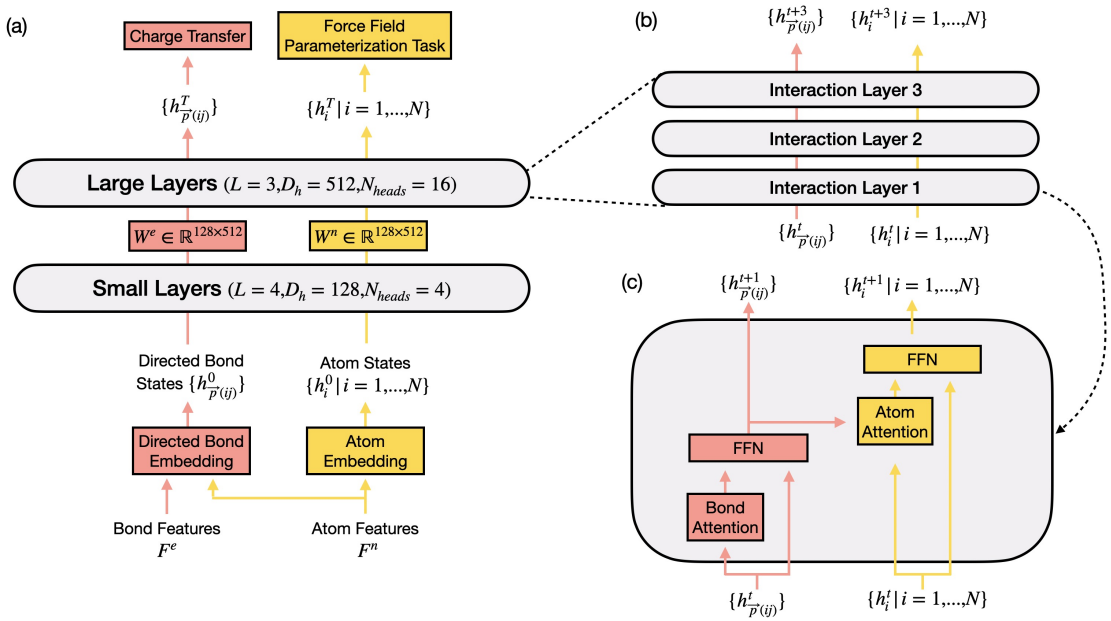

Figure 4.2: **Molecule processing model: (a)** The model to process molecules follows the idea in D-GATs but with hierarchical structure. L is the number of interaction layers, D_h is the dimension of model and N_{heads} is the number of heads in multi-attention mechanism. Between two stacked layers, there exists W^e and W^n to convert the dimension of embeddings. **(b)** The stacked layers consist of several interaction layers. **(c)** Details in interaction layer (FFN refers to feed-forward NNs). Different from D-GATs in Section 3.3, here is no ReadOut function.

FFs are delicate and highly sensitive to their parameters. To augment the expressive power of our GB-FFs models and expand their receptive fields, we employ a hierarchical structure consisting of two stacked layers, namely, the Small and Large Layers. These layers share the same model architecture but operate in different dimensions. Between the two stacked layers, linear transformations (W^e and W^n) are applied to adapt the dimensions of atomic and bond representations.

As shown in Figure 4.2(a), the Large Layers encompass three interaction layers with a model dimension, denoted as D_h, set to 512. These layers serve as the core for detecting chemical environments and generating atomic representations. In contrast, the Small Layers consist of four interaction layers, a model dimension of D_h set to 128 with four attention heads. These layers are primarily utilized for initializing the embeddings and demand relatively fewer computational resources.

Through our experiments, we have found that the utilization of Small Layers leads to a more stable training process by capturing a broader range of atomic environments. Conversely, relying solely on the Large Layers would substantially increase the number of trainable parameters required to achieve a similar receptive field. Furthermore, increasing the dimension D_h of the Large Layers only yields marginal improvements in fitting potential energy and atomic forces. Therefore, our current set of hyper-parameters strikes a balance between computational efficiency and model accuracy.

The output atomic representations are denoted as $\{h^T\} = \{h_i^T | i = 1, ..., N\}$, and the output directed bond representations are noted as $\{h_{\vec{p}(ij)}^T\}$.

4.2.3 Symmetry-Preserving Parameter Generators

The determination of the number of FF parameters depends on the molecule's geometry. Consequently, we break this process down into two distinct parts.

In the first part, the models identify all possible bonds, angles, dihedrals, and non-bonded interaction terms within the molecule's structure. This is achieved through a traversal process. Utilizing RDKit, we enumerate all combinations of bonds, angles, dihedrals, and non-bonded interaction pairs based on the molecular geometry.

In the second part, our model inputs the corresponding atomic representations into the parameter generators for the identified structural elements. However, the parameter generators must ensure atom ordering symmetries. For example, when predicting bond parameters, if we exchange the order of two input atomic representations, the predicted parameters should remain invariant. In contrast to previous similar work, such as Espaloma [28], where equivalent atom permutations were explicitly enumerated, expanding model size and introducing unnecessary operations, we take a different approach. We segment the input atom embeddings based on their intrinsic structure (thus there is no use of the directed bond representations). Linear transformations are then applied to ensure symmetry.

$$h_{r_{ij}} = h_{r_{ji}} = W_r h_i^T + W_r h_j^T \tag{4.5}$$

$$h_{\theta_{ijk}} = h_{\theta_{kji}} = W_{\theta 1} h_i^T + W_{\theta 2} h_j^T + W_{\theta 1} h_k^T \tag{4.6}$$

$$h_{\phi_{ijkl}} = h_{\phi_{lkji}} = W_\phi [h_i^T, h_j^T] + W_\phi [h_l^T, h_k^T] \tag{4.7}$$

$$h_{\varphi_{ijkl}} = h_{\varphi_{jikl}} = h_{\varphi_{jlki}} = h_{\varphi_{ljki}} = h_{\varphi_{ilkj}} = h_{\varphi_{likj}} = W_{\varphi 1} h_k^T + W_{\varphi 2} h_i^T + W_{\varphi 2} h_j^T + W_{\varphi 2} h_l^T \tag{4.8}$$

$$h_{VdW_i} = W_{VdW} h_i^T \tag{4.9}$$

where $[.,.]$ denote concatenation and $h_r, h_\theta, h_\phi, h_\varphi \in \mathbb{R}^{D_h}, W_r, W_{\theta 1}, W_{\theta 2}, W_{\varphi 1}, W_{\varphi 2}, W_{VdW} \in \mathbb{R}^{D_h \times D_h}, W_\phi \in \mathbb{R}^{2D_h \times D_h}$. These embeddings for bond ($\{h_r\}$), angle ($\{h_\theta\}$), torsion ($\{h_\phi\}$), improper torsion term ($\{h_\varphi\}$) and VdW interaction ($\{h_{VdW}\}$) are in the same dimension. And the number of parameters for each term is fixed (for example, one bond term needs two parameters $\{K_r, r_{eq}\}$). According to the corresponding embeddings, we can use the fully connected NNs to predict the FF parameters (see Figure 4.3).

4.2.4 Charge Transfer Model

A complete molecular FF parameterization model must possess the capability to assign atomic charges. Therefore, we have integrated a charge transfer model into our framework to estimate the charges on individual atoms.

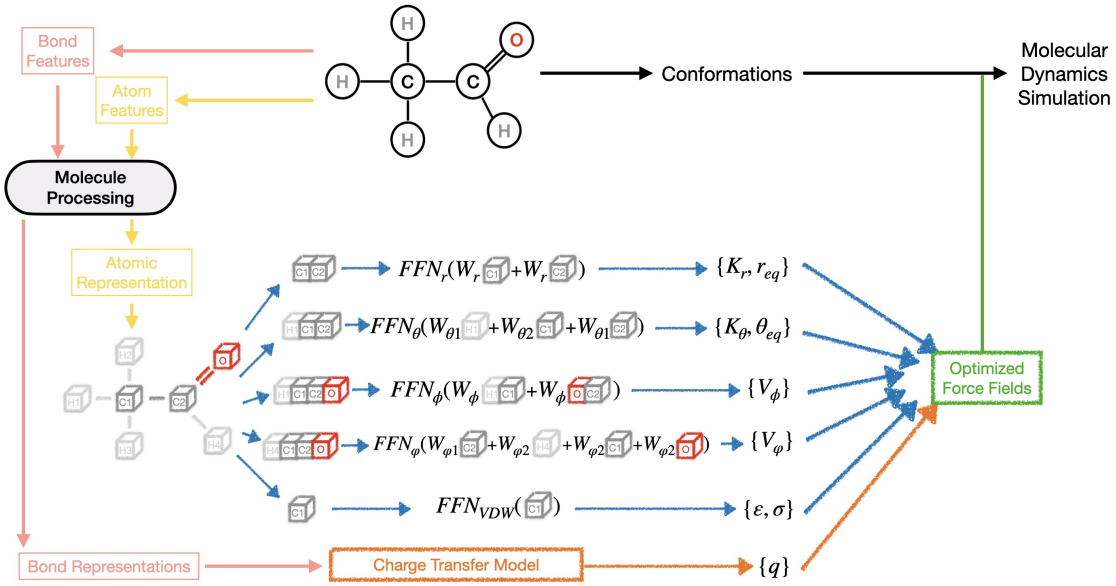

Figure 4.3: **Symmetry-preserving parameter generator:** For a specified molecule, we input atom and bond features to hierarchical D-GATs and obtain the atomic representations and directed bond representations. The symmetry-preserving parameter generators predict all FF parameters, which can be used to do molecular dynamics simulation.

To ensure that the net charge of the molecule aligns with real-world scenarios and to enhance the physical interpretability of charge distribution, we have chosen not to directly predict the charges for each atom using atomic expressions. Instead, we utilize directed bond states $\{h^T_{\vec{p}(ij)}\}$ to estimate the charge transfers between connected atoms. Our molecular processing model, which is based on directed graphs, eliminates the need for additional operations and allows us to predict the charge transfer from one atom to its neighboring atoms.

As illustrated in Figure 4.4, we feed the directed bond features obtained from Figure 4.2 into a feed-forward neural network. This network is responsible for determining the charge transfer in the corresponding bond direction. The final atomic charge is computed by summing the original formal charge and the incoming charges while subtracting the outgoing charges.

While our charge transfer model demonstrates strong performance for small molecular systems, it relies on having prior knowledge of the initial formal charge for each atom. And our current charge transfer model operates with charge movement restricted to connected atoms. This limitation can render our charge transfer model ineffective.

To address this challenge, a potential avenue for future improvements involves adopting the approach proposed by Gilson et al. [250]. This method aims to predict the electronegativity and hardness of each atom, which are defined as the first- and second-order derivatives of potential energy in charge equilibration approaches [59]. Implementing this approach could enable our model to handle cases where only group-level charge information is available, expanding their applicability to larger and more complex molecular systems.

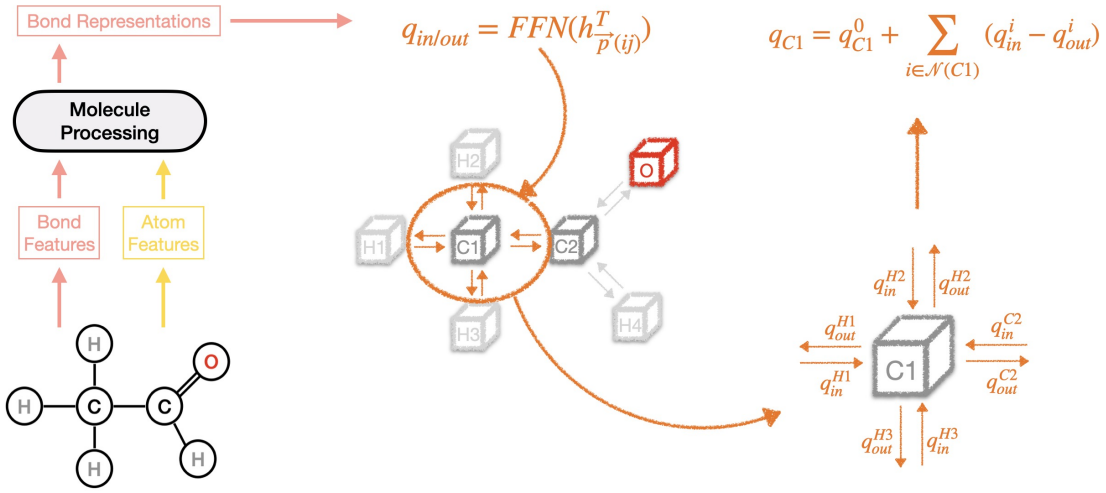

Figure 4.4: **Charge Transfer Model.** The charge is allowed to transfer between connected atoms and the charge in/out is directly calculated by the directed bond embeddings. The final partial charge of atom is the original formal charge plus charge flows in and minus the charge flows out.

4.2.5 Graph-Based Force Fields model

Combining the molecule processing model (Section 4.2.2), the symmetry-preserving parameter generator (Section 4.2.3), and the charge transfer model (Section 4.2.4), we have formed the complete Graph-Based Force Fields (GB-FFs) model. This model serves as a framework for various molecular tasks due to its exceptional capability in detecting chemical environments and providing appropriate atomic/bond fingerprints.

It should be noted that the molecule processing model is initialized by Xavier initialization [110] while the symmetry-preserving parameter generators and charge transfer model are initialized by Kaiming initialization [130].

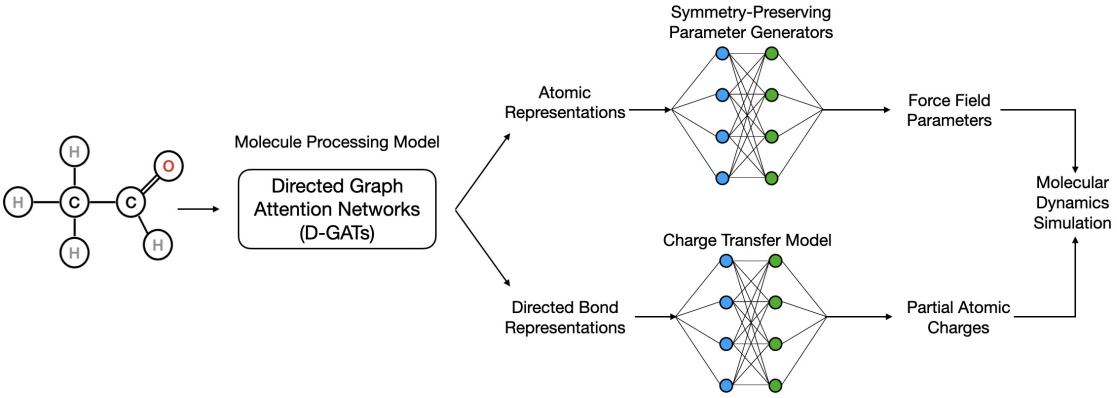

Figure 4.5: **Framework of Graph-Based Force Fields (GB-FFs) Model.** It consists of molecule processing model, the symmetry-preserving parameter generator and charge transfer model.

In this thesis, our focus has been on re-parameterizing GAFF. In fact, our model can be extended to other legacy FFs such as OPLS without modification in scheme. However, extending it to polarizable FFs remains a challenging work that requires further research.

The GB-FFs model encompasses a total of 12,956,944 parameters (about 74 MegaByte). Table 4.2 illustrates the distribution of these parameters among different parts. Our model exhibits efficient runtime complexity, operating at $\mathcal{O}(N)$, with a processing time of less than 0.03 seconds for molecules containing a hundred atoms (tested on a single GPU V100).

GB-FFs Model	Molecule Processing Model	Symmetry-Preserving Parameter Generator	Charge Transfer Model	Total
Number of Trainable Parameters (Proportion)	12,956,944 (70.1%)	4,732,945 (25.6%)	788,482 (4.2%)	18,478,371(100%)

Table 4.2: Trainable parameters in GB-FFs model.

4.2.6 Improving GAFF's Bond Energy Formulation

The harmonic function is basic and often adequate for determining equilibrium geometries in most molecular systems. However, it is not an ideal option for stretch energy because the function value tends to infinity when the bond length is too large, i.e., when the bond is broken. In contrast, the Morse function [251] offers a more precise characterization of bond potential, especially for bonds stretched beyond their equilibrium lengths. To maintain compatibility with GAFF, which utilizes only two bond parameters, namely K_r and r_{eq}, we adopt the following expression for the Morse function:

$$E_{stretching} = \sum_{bonds} \frac{K_r}{\alpha^2}(e^{-\alpha(r-r_{eq})} - 1)^2 \tag{4.10}$$

with $\{K_r, r_{eq}\}$ are the parameters directly from GAFF parameters. α is a new parameter.

The second-order derivative of the bond length (r) for Equation 4.10 is given by $\frac{d^2E_{bonds}}{dr^2} = \sum_{bonds} 2K_r$, which is identical to the second-order derivative of the bond length for the harmonic function in Function 4.2.

In this stage, we aim to investigate the impact of the functional form on FFs. However, the Morse function introduces an additional parameter denoted as α. As a result, we designate α as a fixed hyper-parameter, and maintain the same number of parameters as original GAFF. We have carefully assessed the characteristics of Morse function and its overall performance across various values of α. After thorough evaluations, we find that $\alpha = 2$ yields the best performance, and that our final choice.

Employing the Morse function to approximate stretch energy aligns with the physical laws. As two atoms approach each other closely (i.e., $r \to 0$), the bond energy exhibits a rapid increase. Conversely, as the bond length becomes excessively long, the stretch energy stabilizes at a certain energy level.

4.2.7 Urey-Bradley Terms

Legacy molecular FFs typically adopt function forms based on a balance between computational cost and approximation effectiveness. Although the harmonic function yields only moderate accuracy, it offers high computational efficiency, enabling simulations of larger and more complex systems. Nevertheless, the advancement in computer capacity enables us to introduce supplementary correction terms and integrate additional parameters into conventional FFs, thus

augmenting simulation accuracy. In this subsection, we make some preliminary attempts in this direction.

In this stage, we are moving beyond the constraint of the number of parameters. We start by replacing the harmonic function with the complete Morse function (as described in Function 4.10) to model the stretch energy. $\{K_r, \alpha, r_{eq}\}$ are the FF parameters, initialized as $\{K_r, 2, r_{eq}\}$ and r is the bond length. K_r and r_{eq} are directly from GAFF.

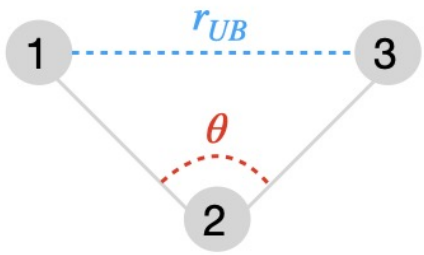

Figure 4.6: **Urey-Bradley term:** The 1-3 endpoints distance r_{UB} is taken into consideration.

The Urey-Bradley (UB) terms [91] serve as a cross-term addressing 1-3 non-bonded interactions that are not adequately covered by bond and angle terms (as illustrated in Figure 4.6). The inclusion of UB terms significantly enhances the accuracy of reproducing subtle nuances in the vibrational spectrum. Nowadays, some FFs, such as CHARMM and AMOEBA, still use UB terms while AMBER and GAFF do not. However, even within the CHARMM General Force Field (CGenFF), developers have adopted a strict policy against introducing new UB terms due to several inherent drawbacks:

Increased Complexity in Parameterization: The addition of UB terms introduces extra complexity to the FF parametrization process, leading to increased underdetermination. Incorporating UB terms requires an extra fitting task to determine optimal parameters, which can be excessively laborious.

Limited Transferability of UB Terms: The 1-3 distance can vary significantly for the same central atom in the same hybridization state, making it challenging to define a universally applicable set of UB parameters.

In the context of general FFs for organic molecules, the second disadvantage mentioned above outweighs all other considerations. However, in the case of our model, the procedure for acquiring atomic representations and predicting FF parameters is automated and efficient. Furthermore, our model provides parameters based on the atomic chemical environment, significantly alleviating concerns associated with low transferability. Consequently, we reintroduce the previously "abandoned" UB terms into GAFF.

Typically, UB terms employ a harmonic function, which is not enough in representing complex interactions adequately. To address this limitation, we introduce a novel functional form and incorporate the UB terms:

$$E_{UB} = \sum_{angles} K_{UB}((\frac{r_{UB_{eq}}}{r_{UB}})^2 - 1)^2 \tag{4.11}$$

with $\{K_{UB}, r_{UB_{eq}}\}$ are the new FF parameters. Given that we lack reference data for $\{K_{UB}, r_{UB_{eq}}\}$, we initialize K_{UB} to 0.1 (in Kcal/mol) and $r_{UB_{eq}}$ (in Å) is initialized using equilibrium structure

parameters r_{eq} and θ_{eq} from GAFF. For instance, in Figure 4.6, the equilibrium bond lengths for bonds 1-3 and 2-3 are denoted as r_{eq}^{12} and r_{eq}^{23}, while the equilibrium angle is indicated as θ_{eq}^{123}. We anticipate that the UB term attains its minimum energy when the structure is in an equilibrium state. According to the Law of Cosines, $r_{UB_{eq}}$ is initialized as follows:

$$r_{UB_{eq}} = \sqrt{{r_{eq}^{12}}^2 + {r_{eq}^{23}}^2 - 2r_{eq}^{12}r_{eq}^{23}\cos(\theta_{eq}^{123})} \tag{4.12}$$

Indeed, it is possible to introduce additional correction terms, such as separating the endpoint distance of the torsion term from the VdW terms and incorporating it as a part of the torsion term. However, this strategy introduces the risk of overfitting and may potentially compromise the performance of the original FFs, obeying our primary objective of optimizing FFs. Consequently, we have limited our modifications to the stretch and bending terms for now.

4.3 Experiments and Results

In this section, we will outline the training strategies and showcase the performance of the GB-FFs model across various databases. It's important to emphasize that in this section, we focus on evaluating our parameters against the GAFF parameters, and we are not comparing them to any other models. Our primary objective here is to assess the effectiveness of optimizing FF parameters through ML methods. We are interested in the achievable improvement brought by our model. And the purpose of this section is not to demonstrate that our model outperforms other FFs.

4.3.1 Atom Type Prediction

Before proceeding to train our model using potential energy and forces, we first utilize the molecule processing model to derive atomic embeddings for reconstructing GAFF atom types.

SPICE (Small-Molecule/Protein Interaction Chemical Energies) [14] is a collection of quantum mechanical data to explore all areas of configuration space that are typically encountered in simulations. It encompasses 15 elements (H, C, N, O, F, P, S, Cl, Br, I, Li, Na, Mg, K, Ca) and a diverse array of chemical groups. There are more than one million conformations in SPICE.

As SPICE encompasses a broader range of elements and exhibits complex molecular structures, we choose it to conduct this testing. The results of this process are presented in Table 4.3.

SPICE	H	C	N	O	P	S	F	Cl	Br	I	Total
Accuracy	99.80%	99.02%	98.87%	99.51%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.42%
Number of Atoms	28,270	24,268	4,685	4,094	86	916	761	473	201	51	63,805

Table 4.3: **Accuracy of the predicted atom types on SPICE.**

Overall, the predictive accuracy is remarkably high, averaging at 99.42%. Particularly, for elements other than H, C, N, and O, the prediction accuracy reaches a perfect 100%. However, when we investigate the misclassification of atomic types, we observe that the primary source of errors stems from handling special structures.

Taking Figure 4.7 as an example, our model incorrectly identifies "cu" as "c2" and "cx" as "c3". These misclassifications are primarily due to the model's difficulty in correctly recognizing triangular systems ("c3" vs. "cx" and "c2" vs. "cu"). Similarly, for carbon atoms, our model sometimes struggles to distinguish square systems ("c3" vs. "cy" and "c2" vs. "cv"), biphenyl systems ("ca" vs. "cp"), and non-pure aromatic systems ("ca" vs. "cc") (refer to Table 4.4).

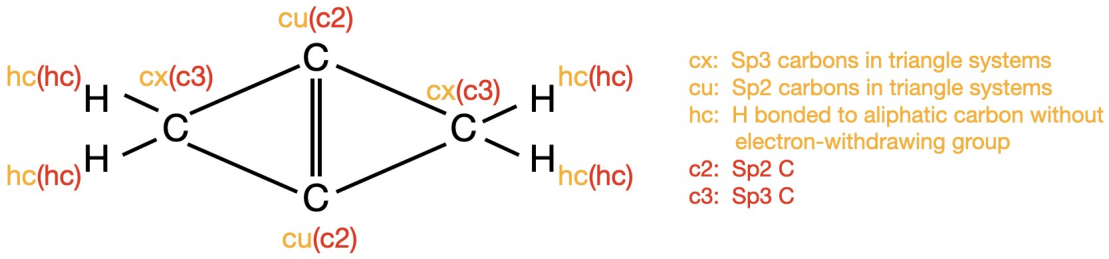

Figure 4.7: **Comparison of atom types by GAFF and our model.** The atom type in orange are from GAFF, while the red are predicted by our model. The incorrect assignment of atom types is primarily due to the failure to recognize triangular systems.

Here we have provide detailed results specifically for carbon atoms. The comprehensive results for all elements can be found in the supplementary results (Section 4.5, Figure 4.15). While the predictions may not always be accurate for these special atomic systems, it is important to recognize that such systems are relatively rare and uncommon in molecular structures. Furthermore, even in cases where our model may not correctly identify these specific systems, it has successfully captured the essential atomic features. This capability enables us to proceed with confidence in predicting FF parameters, which are critical for modeling molecular interactions and simulating molecular systems accurately.

4.3.2 Pre-training on ANI-1 Database

To enhance our model's robustness and performance, it is pre-trained on the ANI-1 database [7, 8] which is, up-to-date, the largest database of Density Functional Theory (DFT) computations for available small organic molecules [7]. This database comprises over 20 millions off-equilibrium conformations of 57,462 small organic molecules extracted from the GDB database [9, 10]. The conformations are built through exhaustive sampling of a subset of the GDB-11 database containing molecules with between 1 to 8 heavy atoms and considering only the species H, C, N and O. The electronic calculations and structure calculations are carried out with the ωB97x [11] density functional and the 6–31 G(d) basis set [12] making it a prime candidate for evaluating ML-driven FF parameterization precision [11].

Before the pre-training phase, it is essential to scale the parameters appropriately. For instance, consider the bond parameters [K_r (in Kcal/mol/Å^2), r_{eq} (in Å)] for the bond type "c-c", which are given as [224.4, 1.5480], with "c" representing the atom type for Sp2 carbon in a carbonyl group. In our model, we aim to predict the corresponding values of K_r and r_{eq}. However, the value of the force constant K_r (224.4) is hundreds of times larger than the equilibrium length r_{eq} (1.5480). Consequently, we need to reduce the force constant K_r in Equation (4.1) by a factor of 100. Similarly, the force constant for angles, K_θ, needs to be reduced by a factor of 10, and the AM1-BCC partial charges should be increased by a factor of 10.

During the pre-training stage, there are three steps, and each subsequent step's initial model is based on the best model saved from the previous step.

1. Training the GB-FFs model to give the same parameters as GAFF and the loss function is defined as Equation (4.13), where $\texttt{MSE}(P_X)$ denotes the mean square error calculated by the difference of X (bond or angle or dihedral or VdW or charge) parameters from GB-FFs

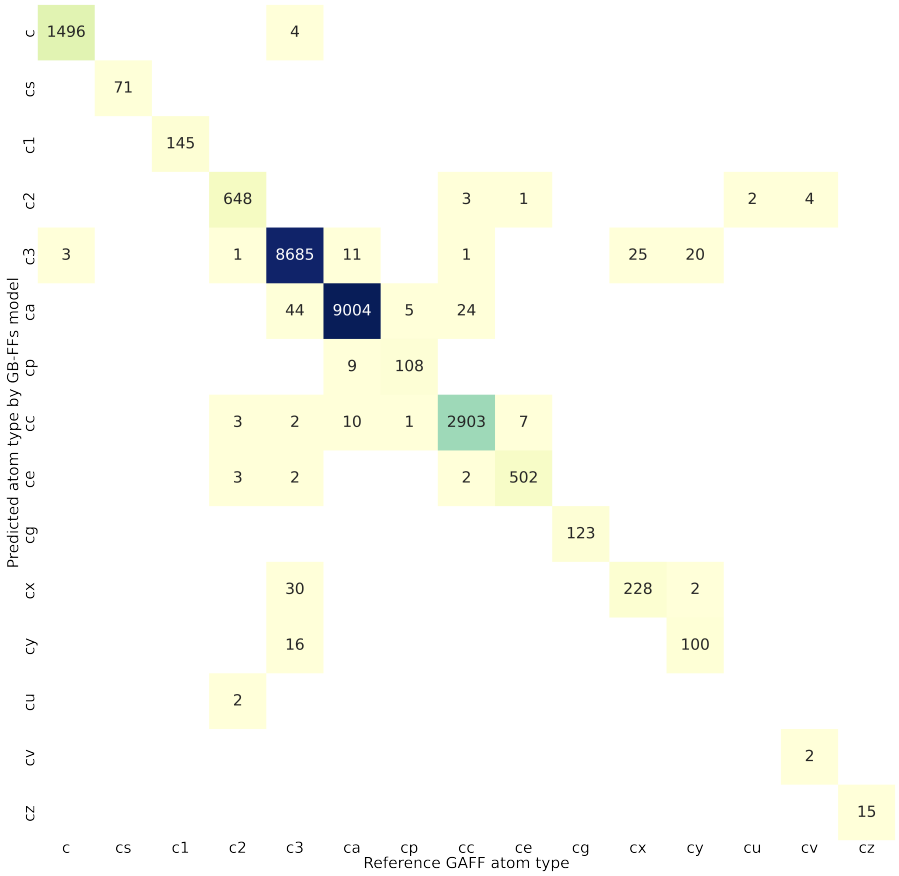

Table 4.4: **Predicted atom types for carbon.** The numbers in the table represent the frequency for each case.

model and GAFF parameters. The models are trained for 10 epochs.

$$\texttt{Loss} = \texttt{MSE}(P_{bonds}) + \texttt{MSE}(P_{angles}) + \texttt{MSE}(P_{dihedrals}) + \texttt{MSE}(P_{charges}) + \texttt{MSE}(P_{VdW}) \quad (4.13)$$

2. Training the GB-FFs model to give the same parameters and the same energy as GAFF. The loss function is defined as Equation (4.14), where $\texttt{MSE}(E_X)$ denotes the mean square error of energy calculated by the X (bond or angle or dihedral or VdW or charge) parameters from GB-FFs model and the energy calculated by the GAFF parameters. It should be noted that the energy in this step indicates the energy for each interaction, i.e. for every bond/angle/dihedrals and VdW/charge interaction pair. The models are trained for 10

Number of layers	4/3	Dimension of model	128/512	Initial learning rate	5E-04	Training conformations	1.76E7
Number of heads	4/16	Minimum heavy atoms	1	Smallest learning rate	1E-06	Validation conformations	2.17E6
Number of atom features	38	Maximum heavy atoms	8	Gamma	0.999	Test conformations	2.17E6
Number of bond features	12	Dropout	0.1			Batch size	256

Table 4.5: Details of pre-training for GB-FFs model on ANI-1 database.

epochs.

$$\begin{aligned}\texttt{Loss} =&\texttt{MSE}(P_{bonds}) + \texttt{MSE}(P_{angles}) + \texttt{MSE}(P_{dihedrals}) + \texttt{MSE}(P_{charge}) + \texttt{MSE}(P_{VdW}) \\ &+\texttt{MSE}(E_{bonds}) + \texttt{MSE}(E_{angles}) + \texttt{MSE}(E_{dihedrals}) + \texttt{MSE}(E_{charge}) + \texttt{MSE}(E_{VdW})\end{aligned} \tag{4.14}$$

3. Training the GB-FFs model to give the same parameters and the same total potential energy as GAFF. The loss function is defined as Equation (4.15), where $\texttt{MSE}(\Delta E_{total})$ denotes the mean square error of relative energy calculated using the parameters from GB-FFs model and the total energy calculated by the GAFF parameters. The relative energy refers to the energy difference between a given conformation and the conformation with lowest energy. By focusing on ΔE, we can investigate the impact of different conformations on the potential energy, while the absolute energy is influenced by the system's conditions. Our model is capable of predicting partial charge. In the pre-training stage, our charge is expected to approach AM1-BCC charge but we still use the AM1-BCC charge (other parameters are given by GB-FFs model) to compute the potential energy. There are also L2 regularization for dihedral parameters and the transfer charge (not the partial charge). Without the regularization loss, dihedral terms give unreasonable results. And the relatively small transfer charge corresponds to the real situation. ANI-1 database focus on the intramolecular interactions and the long-range interactions are not evident. We assign a weight of 100 to the loss related to VdW parameters and charges to ensure that Gb-FFs model exhibits a comparable performance to the original GAFF when it comes to non-bonded interactions. The models are trained for 50 epochs the only the models with smallest validation error will be stored.

$$\begin{aligned}\texttt{Loss} =&\texttt{MSE}(P_{bonds}) + \texttt{MSE}(P_{angles}) + 10 * \texttt{MSE}(P_{dihedrals}) + 100 * \texttt{MSE}(P_{charge}) \\ &+100 * \texttt{MSE}(P_{VdW}) + 0.1 * \texttt{L2}(P_{dihedral}) + 0.1 * \texttt{L2}(charge_{transfer}) \\ &+\texttt{MSE}(\Delta E_{potential})\end{aligned} \tag{4.15}$$

In Function 4.2, an equilibrium state exists for both bond and angle terms, wherein we apply the harmonic function (or Morse function in our GB-FFs Morse model and GB-FFs UB model). Additionally, for the VdW interactions, we employ the Lennard-Jones function and it also denotes an equilibrium distance. It's important to highlight that the partial potential energy reaches its minimum at these equilibrium states.

The FF serves as an empirical tool, with parameters meticulously tuned to accommodate conformations around these equilibrium states. Consequently, FFs typically exhibit superior performance when applied to conformations close to these equilibrium states. Conformations that deviate significantly from these equilibrium states introduce a risk of inaccuracy due to extrapolation. It is necessary to filter these conformations to make our model more robust.

However, in the context of pre-training on the ANI-1 database, the primary objective is to enable the model to learn more about molecular features. Consequently, we set the filtering threshold relatively high to retain as much pre-training data as possible.

In our approach, if a conformation exhibits more than one bond energy or angle energy greater than 100 Kcal/mol, or if it has more than one non-bonded atomic pair with a VdW energy greater than 50 Kcal/mol, we discard that specific conformation. Because the energy depends on FF parameters, the filtering process is dynamic since the parameters provided by the ML model change after each update. After predicting the FF parameters using the GB-

ANI-1 Database	GAFF (AM1-BCC charge)	GB-FFs GAFF (AM1-BCC charge)	GB-FFs Morse (AM1-BCC charge)	GB-FFs UB (AM1-BCC charge)
RMSE for total energy (Kcal/mol)	21.6031	12.6951	3.7845	2.9399
Number of conformations (Original: 21,940,262)	21,140,785	20,698,339	21,581,856	21,672,865

Table 4.6: Pre-training results of GB-FFs model on ANI-1 test database.

FFs model, we calculate the corresponding energy for each conformation and then remove those that significantly deviate from equilibrium states. The remaining conformations are used for calculating the loss function. This dynamic filtering process ensures that only conformations close to equilibrium are considered for loss calculation, thereby enhancing the accuracy and effectiveness of the model. On average, around 2% of conformations are filtered in each epoch.

There are originally 57,462 molecules in ANI-1 database. After processing, 55 molecules fail to generate GAFF parameter files, leaving 57,407 molecules for further analysis. These molecules are randomly divided into training/validation/test set, following an 8:1:1 ratio.

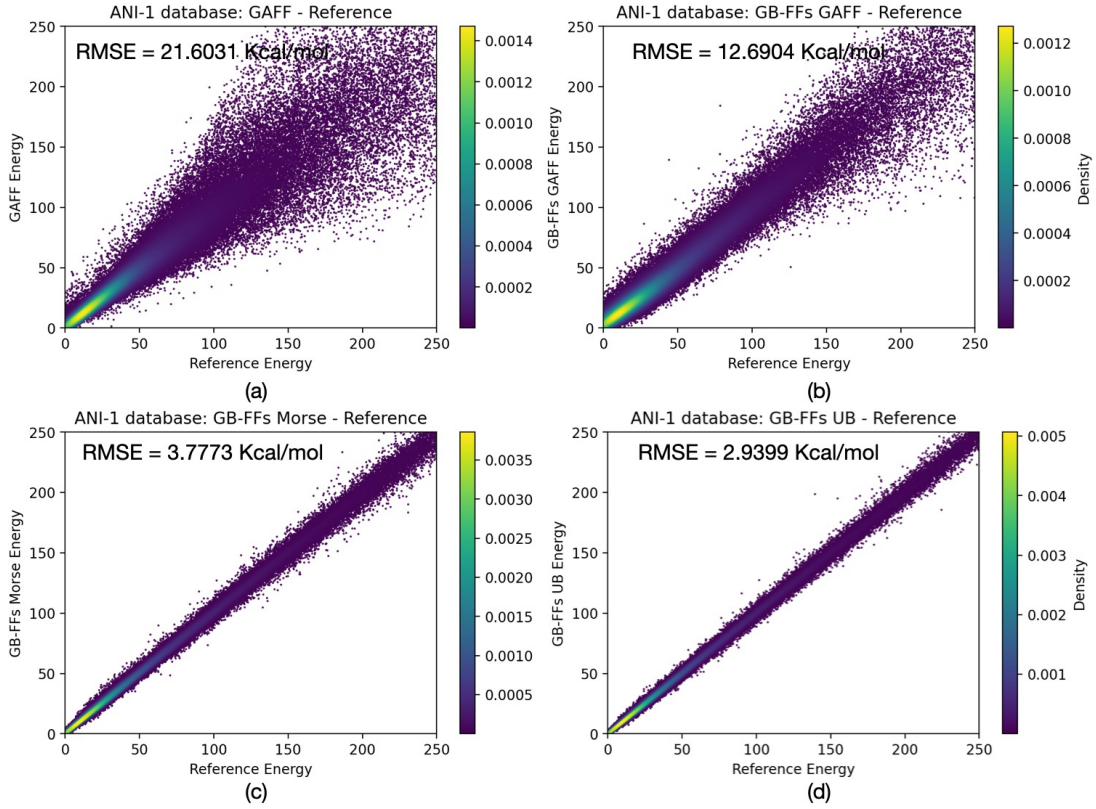

Figure 4.8: **Predicted energy v.s. reference energy on ANI-1 test dataset.** **(a)**Original GAFF **(b)**GB-FFs GAFF **(c)**GB-FFs Morse **(d)**GB-FFs UB

As presented in Section 4.2.6 and 4.2.7, there are four models:

1. Original GAFF, noted as "GAFF".

2. GAFF functions with parameters generated by GB-FFs model, noted as "GB-FFs GAFF".

3. Replacing harmonic function by Morse function to estimate bond energy, noted as "GB-FFs Morse" (Section 4.2.6).

4. Using the Morse function with three parameters and adding the UB terms, noted as "GB-FFs UB" (Section 4.2.7).

The results of various models on the test set are presented in Table 4.6 and Figure 4.8. It is important to note that we calculate errors in terms of relative energy (the energy difference between minima) rather than absolute energy to avoid considering heats of formation.

From the results presented in Table 4.6, it is evident that the FF parameters provided by the GB-FFs model exhibit superior performance compared to the original GAFF. However, it is crucial to emphasize that the original GAFF was never trained on the ANI-1 database, so its lower performance is expected.

We have also observed the significant impact of function forms on the performance of FFs. When we replace the harmonic function with the Morse function to approximate the potential energy associated with chemical bonds, the root mean square error (RMSE) of the potential energy decreases from 12.7 to 3.8 Kcal/mol. Furthermore, when we incorporate UB terms for bending energy, the error is further reduced. This underscores the fact that optimizing FFs necessitates not only data-driven approaches such as optimizing NN structures and extracting atomic fingerprints but also considerations of mathematical and chemical perspectives to employ function forms that better simulate potential energy and align with actual conditions.

4.3.3 Fine-tuning on SPICE and DES370K Databases

SPICE (Small-Molecule/Protein Interaction Chemical Energies) [14] is a collection of quantum mechanical data aimed at training potential functions and atomic forces. Its primary emphasis lies in simulating interactions between drug-like small molecules and proteins. We have chosen to leverage the SPICE database to fine-tune our GB-FFs model for several compelling reasons:

- It covers a wide range of chemical space: SPICE encompasses 15 elements (H, C, N, O, F, P, S, Cl, Br, I, Li, Na, Mg, K, Ca) and a diverse array of chemical groups. It includes charged and polar molecules, as well as neutral ones. The intention is to sample a broad spectrum of covalent and non-covalent interactions.

- It covers a wide range of conformations: SPICE incorporates both low and high energy conformations (more than one million conformations). It is designed to explore all areas of configuration space that are typically encountered in simulations.

- It is made up of a collection of subsets (including dipeptides, solvated amino acids, PubChem [19], monomer and dimer [20], ion pairs) and each one is designed to provide a particular type of information.

- It includes forces as well as energies: Unlike many databases that only provide energies, SPICE also includes forces. This inclusion significantly augments the information content of the database.

- The computations are performed at the ωB97M-D3(BJ) functional [15, 16] and def2-TZVPPD basis set [17, 18], which is known for its high accuracy.

- It is freely available under a non-restrictive licence.

DES370K [20] is a collection of dimer interaction energies computed using the high-level coupled-cluster singles and doubles with perturbative triples (CCSD(T)) [252] method at the complete basis set (CBS) [253] level of theory. This extensive database encompasses 370,959 unique geometries. It includes 392 monomers, encompassing both neutral molecules and ions, ranging from water to the functional groups commonly found in proteins. The complexes in this database represent various interaction motifs, including electrostatic-dominated (Hydrogen bonding), dispersion-dominated, and mixed (electrostatic/dispersion) interactions.

A notable subset within the DES370K database comprises QM-optimized dimer structures, which served as the starting points for generating additional structures along one-dimensional radial profiles. In order to augment orientational diversity and ensure thorough sampling of the internal degrees of freedom within larger chemical species, the database also incorporates a substantial ensemble of structures, along with their corresponding radial profiles, obtained from MD simulations.

The SPCIE database primarily concentrates on quantifying energy and forces arising from intermolecular interactions. In contrast, the DES370K database specifically addresses non-bonded interaction energies between two monomers, with a particular focus on dispersion and electrostatic interactions. The simultaneous utilization of these two datasets for fine-tuning ensures that GB-FFs model upholds a high accuracy in simulating both intra-molecular and intermolecular interactions.

As GAFF includes only H, C, N, O, F, P, S, Cl, Br, and I, we exclude the molecules containing elements Li, Na, Mg, K, Ca. In the SPICE database, there are originally 19,238 compounds. After filtering out compounds containing metal atoms and those unable to generate GAFF parameter files, 18,613 compounds remained. In the DES370K database, there are originally 10,776 compounds. After filtering out compounds containing metal atoms and those unable to generate GAFF parameter files, 12,669 compounds remained. These compounds are then randomly divided into training/validation/test sets, following an 8:1:1 ratio.

The loss function is defined as Equation (4.16). The atomic force is combined into the loss function to make the parameters given by our GB-FFs model be corresponded with realistic situation. The high loss weight assigned to VdW parameters and charges reflects the sensitivity and fragility of long-range interactions. Our objective is to preserve their performance to match that of the original GAFF. In the DES370K database, where the primary focus lies on non-bonded interaction energy, we have allocated a substantial weight to $\texttt{MSE}(\Delta E_{DES370K})$ for the same reason – ensuring model's accuracy and fidelity in capturing these critical interactions.

$$\begin{aligned}\texttt{Loss} =& \texttt{MSE}(P_{bonds}) + \texttt{MSE}(P_{angles}) + 10 * \texttt{MSE}(P_{dihedrals}) + 100 * \texttt{MSE}(P_{charge}) \\ &+1000 * \texttt{MSE}(P_{VdW}) + 0.1 * \texttt{L2}(P_{dihedral}) + 0.1 * \texttt{L2}(charge_{transfer}) \\ &+\texttt{MSE}(\Delta E_{SPICE}) + 100 * \texttt{MSE}(\Delta E_{DES370K}) + 10 * \texttt{MSE}(\Delta F_{SPICE})\end{aligned} \tag{4.16}$$

The filtering rule is as follows: if a conformation exhibits more than one bond energy or angle energy exceeding 50 Kcal/mol, or if it involves more than one non-bonded atomic pair with a Van der Waals (VdW) energy greater than 10 Kcal/mol, it is deemed unsuitable and discarded. On average, approximately 4% of conformations are filtered out during each epoch.

The criteria for evaluating our models comprise the sum of RMSE in energies and atomic forces, denoted as $\texttt{RMSE}(\Delta E_{SPICE}) + 100 * \texttt{RMSE}(\Delta E_{DES370K}) + 10 * \texttt{RMSE}(\Delta F)$. The models are trained for 500 epochs and early-stopping is applied (if the criteria value on validation dataset ceases to increase for 30 consecutive epochs, the training stage is done).

To illustrate the advantages of pre-training, let us consider the "GB-FFs GAFF" model. After pre-training on the ANI-1 database, we fine-tune the embedding components in the molecule processing model (refer to Figure 4.9). The ANI-1 database exclusively contains H, C, N, and O, with atomic formal charges consistently set to zero. In contrast, the SPICE and DES370K databases encompasse P, S, F, Cl, Br, I, and may possess non-zero formal charges.

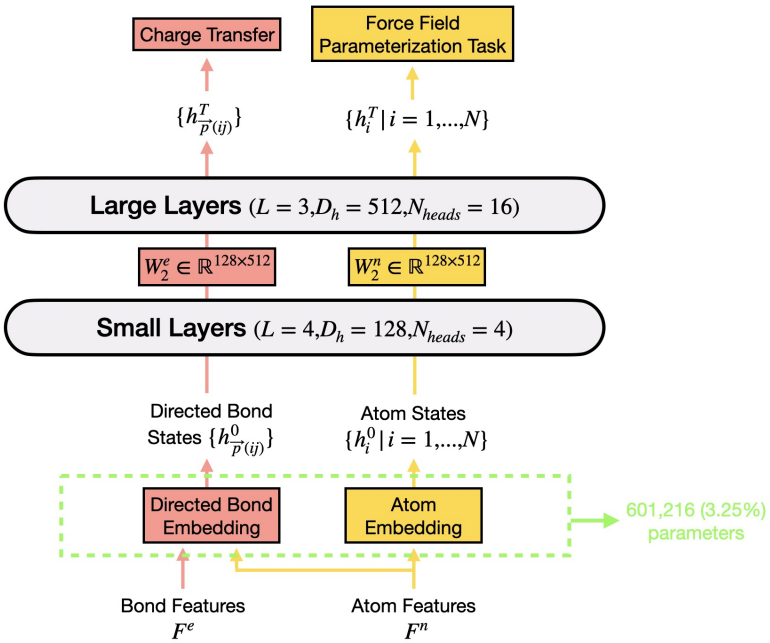

Figure 4.9: **Embedding layers in molecule processing model.** Fine-tuning the embedding layers (in green box), which contains thousands of parameters (about 3.25% of total GB-FFs model parameters)

The GB-FFs model can be generalized to new databases by only fine-tuning the embedding layers, which represent only 3.25% of the total trainable parameters. To illustrate this, consider the "GB-FFs GAFF" model presented in Table 4.7. Fine-tuning the model's embedding layers results in improved FF parameters compared to the original GAFF. The remarkable success of transfer learning from the ANI-1 database to new databases underscores the efficacy of pre-training on ANI-1 in capturing the fundamental, intrinsic features of molecules.

We then fine-tune all the pre-trained models discussed in Section 4.3.2. In the case of the "GB-FFs GAFF" model, we consider two scenarios: one with AM1-BCC charges and another with GB-FFs charges. This distinction arises because GAFF provides all FF parameters except for charges. By employing the "GB-FFs GAFF" model alongside AM1-BCC charges, our aim is to demonstrate the superiority of the FF parameters derived from our GB-FFs model over those supplied by GAFF. For the goal of constructing a complete FF, all other models utilize GB-FFs charges.

Simultaneously fitting the potential energy and atomic forces is challenging, particularly when dealing with the diverse compounds contained within the SPICE database. The results are summarized in Table 4.8. In comparison to the original GAFF, our GB-FFs model yields significantly reduced errors in fitting relative potential energies (RMSE reduced from 6.03 Kcal/-mol to approximately 3.0 Kcal/mol), with the RMSE of atomic forces decreasing to less than

		GAFF (AM1-BCC charge)	GB-FFs GAFF Fine-tune Embedding Layers (AM1-BCC charge)	GB-FFs GAFF (AM1-BCC charge)
SPICE Database	RMSE for total energy (Kcal/mol)	6.0312	5.3418	2.9669
	RMSE for force (Kcal/mol/Å)	13.3899	9.2076	5.9483
	Number of conformations (Original: 1,079,834)	1,034,735	1,011,695	1,007,759
DES370K Database	RMSE for interaction energy (Kcal/mol)	1.7395	1.3825	1.1202
	Number of conformations (Original: 323,409)	304,983	311,721	312,048

Table 4.7: Comparison of the results for original GAFF, GB-FFs GAFF fine-tuned on embedding layer and GB-FFs GAFF fine-tuned on all trainable parameters.

6 Kcal/mol/Å(down from 13.39 Kcal/mol/Å). Undoubtedly, these improvements will bring better performance in molecular dynamics simulations. The improvement in DES370K database is limited (RMSE of interaction energy reduced from 1.74 Kcal/mol to about 1.65 Kcal/mol).

Table 4.8 also presents the RMSE between GB-FFs charges and AM1-BCC charges, demonstrating the charge transfer model's capability to predict charge distribution.

		GAFF (AM1-BCC charge)	GB-FFs GAFF (AM1-BCC charge)	GB-FFs GAFF (GB-FFs charge)	GB-FFs Morse (GB-FFs charge)	GB-FFs UB (GB-FFs charge)
SPICE Database	RMSE for total energy (Kcal/mol)	6.0312	2.9669	3.0559	3.0115	2.5990
	RMSE for force (Kcal/mol/Å)	13.3899	5.9483	5.9788	5.3987	4.1565
	RMSE for charge (C)	-	-	0.0441	0.0432	0.0404
	Number of conformations (Original: 1,079,834)	1,034,735	1,007,759	1,007,083	1,007,952	1,018,490
DES370K Database	RMSE for interaction energy (Kcal/mol)	1.7395	1.1202	1.6456	1.6702	1.3388
	RMSE for charge (C)	-	-	0.0644	0.0648	0.0448
	Number of conformations (Original: 323,409)	304,983	312,048	312,417	313,155	286,631

Table 4.8: RMSE for potential energy and force for parameters from different models on SPICE database.

When considering the impact of charges, the calculation of partial atomic charges typically involves quantum chemical computations. Due to computational constraints, we use AM1-BCC charges obtained from a semi-empirical quantum chemical package as a replacement for RESP charges [84, 85]. These charges are then applied in conjunction with the GAFF. However, the computational speed of AM1-BCC charge using the "antechamber" command in AMBER remains significantly slower in comparison to our charge transfer model. To provide a perspective, for a molecule with 50 atoms, AMBER takes 111 seconds to compute AM1-BCC charges, while our models require only 0.018 seconds. And GB-FFs charges demonstrate comparable performance to AM1-BCC charges.

Surprisingly, despite the substantial improvements achieved in fitting potential energy by altering the functional forms (as observed in the "GB-FFs Morse" and "GB-FFs UB" models) on the ANI-1 database, these two models only yield a modest reduction in relative potential energy errors when applied to the SPICE database. We attribute this difference to the nature of the conformations within the two databases. The conformations in ANI-1 database tend to deviate further from the equilibrium state, resulting in a wider range of variations in individual stretching or bending energies. Conversely, the atomic conformations in SPICE database are closer to the equilibrium state. Given that the harmonic function effectively fits conditions near the equilibrium state, the advantages of the Morse function become evident primarily when the system is far from equilibrium—such as when two atoms are either too close or too far apart. Consequently, the "GB-FFs Morse" and "GB-FFs UB" models can offer only limited improvements when compared to the "GB-FFs GAFF" model.

4.3.4 Intermolecular Interaction Accuracy: S66×8 benchmark

The benchmark used in the study of intermolecular interactions is the S66×8 database,[254] which comprises 66 dimers positioned at 8 distinct intermolecular distances, resulting in a total of 528 unique structures.

Both DES370K and S66×8 concentrate on the intermolecular interactions. The S66×8 database is a widely known reference database for assessing the accuracy of various computational methods from FFs to dispersion models [255–257].

For these systems, reference interaction energies are computed using the high-level coupled-cluster singles and doubles with perturbative triples (CCSD(T)) method [252] at the complete basis set (CBS) level of theory [253]. The complexes in the database represent various interaction motifs, including electrostatic-dominated (hydrogen bonding), dispersion-dominated, and mixed (electrostatic/dispersion) interactions.

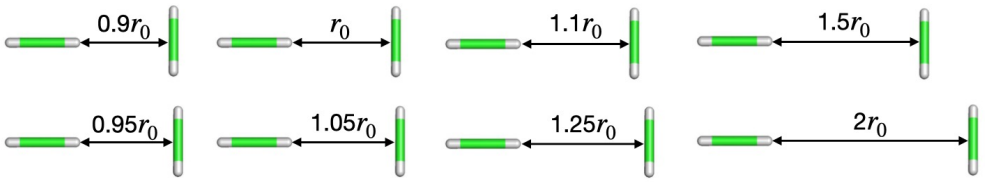

Figure 4.10: **Distance of monomers in S66 × 8 database.** Example of the dimers (Ethyne-Ethyne) at eight distinct intermolecular distances.

To construct the S66×8 dataset, the authors scaled the closest intermolecular distance in the dimers along an intermolecular axis. The axis definition varies for different types of complexes, and detailed information regarding the displacement coordinate can be found in their article [254], particularly in the Supporting Information, Table S3. To generate the dataset, one of the monomers was moved along this axis to achieve minimum distances at 0.9, 0.95, 1.0, 1.05, 1.1, 1.25, 1.5, and 2.0 times the equilibrium value (refer to Figure 4.10).

It is important to note that during these variations in intermolecular distance, the monomers' geometries remained fixed, thus excluding the consideration of deformation energies of the monomers. This design allows the database to specifically evaluate the accuracy of intermolecular interactions.

ANI-1 and SPICE primarily focus on intra-molecular potential energy components such as stretching energy, bending energy, and dihedral energy. To improve the fitting of intermolecular

S66 × 8 Database	GAFF (AM1-BCC charge)	GB-FFs GAFF (AM1-BCC charge)	GB-FFs GAFF (GB-FFs charge)	GB-FFs Morse (GB-FFs charge)	GB-FFs UB (GB-FFs charge)
MAE for interaction energy (Kcal/mol)	0.9368	0.6609	0.6914	0.6376	0.6125
RMSE for interaction energy (Kcal/mol)	1.8388	0.9886	0.9967	0.9191	0.9180

Table 4.9: MARE and RMSE of potential energy on S66 × 8 database.

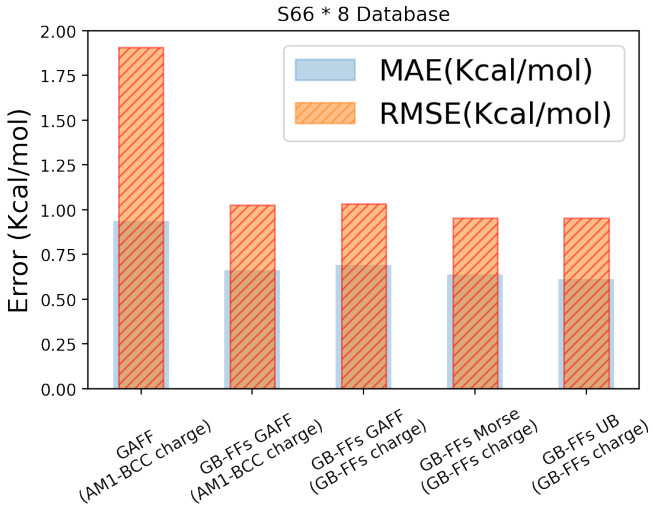

Figure 4.11: **MARE and RMSE of potential energy on S66 × 8 database.**

interactions, specifically VdW interactions and electrostatic potential energy, we have included DES370K as part of the fine-tuning database. The Mean Absolute Error (MAE) and RMSE values are presented in Table 4.9 and Figure 4.11.

In contrast to the original GAFF, which demonstrates an interaction energy RMSE of 1.84 Kcal/mol, the GB-FFs models have the capacity to reduce the RMSE to under 1 Kcal/mol. This experiment proves that the parameters offered by GB-FFs models can indeed guarantee the precision of long-range interactions. Moreover, GB-FFs model introduces an innovative methodology for the development of non-bonded term parameters.

We present four example dimers in Figure 4.12, with full results available in the supplementary materials (Section 4.5, Figure 4.16). Notably, the GB-FFs models have effectively learned the water-water interactions from the SPICE database, which holds significant importance in molecular dynamics simulations within solvation environments. For dimers such as Benzene-Benzene and Ethyne-water, while the GB-FFs models may not yield perfect FF parameters, they do capture the trends in potential energy variations that correspond to changes in reference energies. However, it is worth mentioning that for the water-$MeNH_2$ dimers, our models perform even worse than the original GAFF.

4.3.5 Performance Assessment on Torsion Energy

Torsion energies play a crucial role in both biological and small molecular systems. However, accurately assessing torsional parameters within FF is a formidable challenge due to their reliance

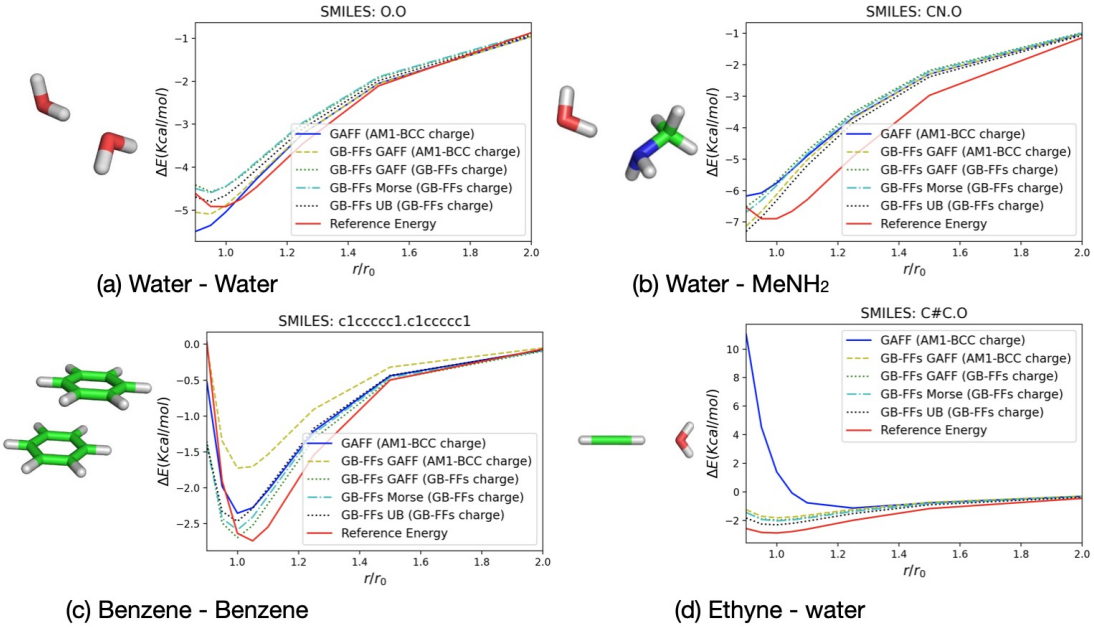

Figure 4.12: **Results of four example on S66 × 8 database.**

on computationally expensive calculations and complicated fitting procedures. Moreover, these parameters are highly sensitive to the local chemical environment, reducing their transferability across different molecular systems. Consequently, they often rely on simplified transferability rules, which can result in inaccuracies.

Achieving accurate torsion profiles without the need for extensive torsion fitting is of great importance in FF parameterization. Our models are fine-tuned on SPICE database by fitting energy and atomic forces. We evaluate the performance of GB-FFs parameterization and compare it to GAFF parameterization on a highly accurate torsion scan database [258].

This database comprises 62 fragments containing drug-like functional groups, along with their CCSD(T) [252]/CBS [253] single-point energies calculated on optimized geometries employing MP2 [259, 260]/6-311+G** [261, 262].

For each molecule within the 62 fragments, a specific dihedral angle is systematically varied from -170° to 170° in 10° increments (the chosen dihedral angle for modification is indicated in the supplementary results, Section 4.5, Figure 4.17). The corresponding changes in relative energy are recorded as the reference energy.

Torsion Scan Database	GAFF (AM1-BCC charge)	GB-FFs GAFF (AM1-BCC charge)	GB-FFs GAFF (GB-FFs charge)	GB-FFs Morse (GB-FFs charge)	GB-FFs UB (GB-FFs charge)
MAE for energy (Kcal/mol)	1.9693	0.6787	0.8262	0.8789	0.6468
RMSE for energy (Kcal/mol)	3.5351	1.0693	1.3369	1.3677	0.9955

Table 4.10: RMSE and MAE for potential energy on torsion scan database.

The results are presented in Table 4.10 and Figure 4.13. In comparison to the original GAFF, the GB-FFs models fine-tuned on the SPICE and DES370K databases yield FF parameters that

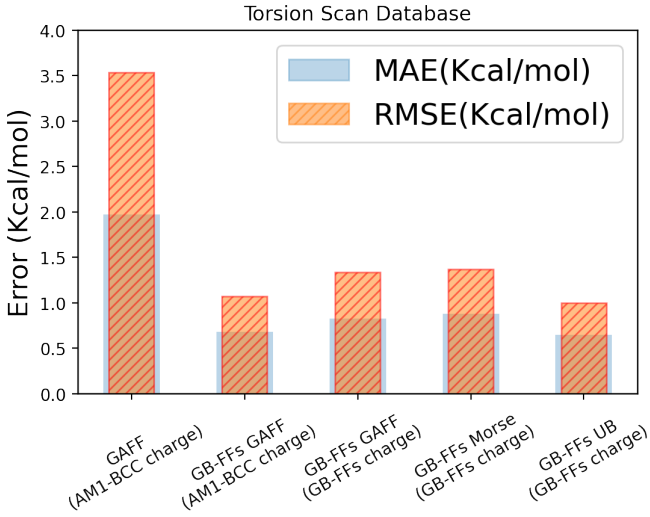

Figure 4.13: **RMSE and MAE for potential energy on torsion scan database.**

offer a better fitting for the potential energy variations arising from dihedral angle changes. Although the performance of different models may vary (refer to Figure 4.14 (a) and (b)), the errors are in the same level. This suggests that the fine-tuned GB-FFs models, whether utilizing AM1-BCC or GB-FFs charges and regardless of whether functional forms are modified or not, can effectively predict energy changes resulting from dihedral angle variations. In some instances, although there may be a gap between our model's predictions and the reference energies (see Figure 4.14 (c) and (d)), the trend of the changes aligns with the actual behavior.

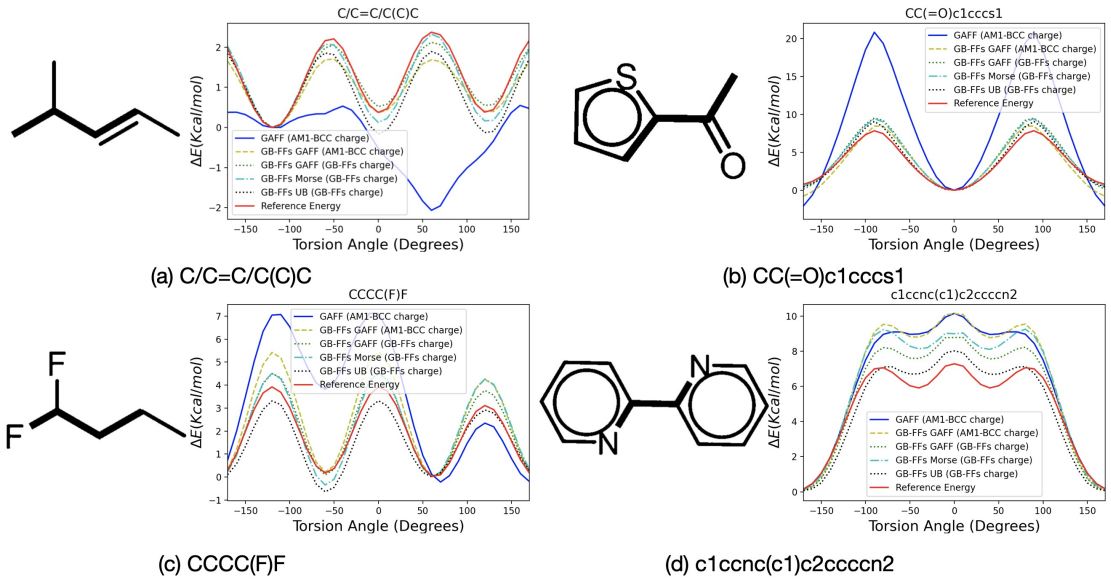

Figure 4.14: **Results of four examples on torsion scan database.**

This assessment aims to highlight the capabilities of the GB-FFs model in accurately capturing torsional energies.

4.4 Conclusions

In this chapter, we have explored the challenges associated with Force Fields (FFs). While molecular dynamics simulations are powerful tools for studying molecular systems, their applicability has often been constrained by the accuracy of the employed FFs. To address this limitation and move away from relying on atom types, we have introduced the Graph-Based Force Fields (GB-FFs) model for optimizing FF parameterization.

Our primary objective is to enhance the General Amber Force Field (GAFF) by using a graph neural network (GNN), called Directed Graph Attention Networks (D-GATs). These networks are powerful in processing directed molecular graphs and extracting atomic fingerprints. These atom-level representations are subsequently fed into a parameter generator to produce the corresponding FF parameters. Typically, GAFF recommends the use of RESP charges. However, when dealing with a large amount of data, for computational efficiency, AM1-BCC charges can serve as a viable alternative. We employ a charge transfer model to calculate the final partial atomic charges, with AM1-BCC charges serving as the target values.

Given the complexity and sensitivity of FF parameterization, we employ a two-step approach. First, we pre-train GB-FFs models on the ANI-1 database, followed by fine-tuning on the SPICE and DES370K databases. This enhances the robustness and fitting capabilities of neural networks. It is essential to note that our FF parameterization are based on the GAFF, thus our parameter comparisons are exclusively with GAFF.

We extensively validate our GB-FFs model on multiple databases, demonstrating their effectiveness in capturing inter-molecular interactions (the RMSE of inter-molecular potential energies on the S66×8 database has been reduced from 1.84 Kcal/mol to less than 1.0 Kcal/mol) and energy variations resulting from dihedral angles. On the torsion scan database, the RMSE of potential energies has been lowered from 3.54 Kcal/mol to 1.34 Kcal/mol, and the trends in energy variations correspond to the actual situation.

Our machine learning (ML) parameterization method liberates FF parameters from atom types, relying entirely on the atomic chemical environment. Furthermore, the flexibility of our approach enables its straightforward extension to other non-polarizable FFs. In the case of polarizable FFs, their treatment of charges varies, necessitating specific algorithmic modifications. For example, for AMOEBA, which considers polarizability and requires electrostatic multipole moments, we can incorporate atomic positions in the charge transfer model (as discussed in Section 4.2.4) to derive charge distributions based on molecular geometry. Similarly, we propose utilizing our GB-FFs model to simplify certain steps in the Poltype automatic parameterization procedure [244, 245].

We have also investigated the impact of functional forms on FF performance. On the pre-training database, replacing the harmonic potential with the Morse function for evaluating stretching energies and adding Urey-Bradley terms proves to be highly effective in reducing the RMSE in relative potential energy (from 12.7 to 2.9 Kcal/mol). However, this improvement is less observed on fitting potential energies on the SPICE database. Nonetheless, errors in atomic forces are significantly reduced (from 6.0 to 4.2 Kcal/mol/Å). This difference can be attributed to the fact that molecular conformations in the SPICE database are closer to equilibrium.

In summary, our research addresses some of the limitations of traditional FFs by integrating ML techniques. Through optimizing the parameterization process using GNN, we have significantly enhanced the performance of FFs, leading to improved accuracy and efficiency for

simulating a wide range of molecular systems. These findings open up new possibilities for advancing molecular dynamics simulations and offer a promising method for future researches in this field.

Moving forward, our future directions can be categorized into three key aspects:

Upgrading ML Model Architecture. While our current approach utilizes D-GATs for processing molecules and extracting atomic fingerprints, we aim to further enhance our model architecture. One possibility is exploring higher-order message flows [93], which can efficiently merge information from neighbor atoms, thus improving the model's expressive ability without adding more layers. Additionally, incorporating charge equilibration approaches [59] for assigning partial atomic charges should be expected. Moreover, we can consider integrating atomic positions using equivariant graph attention [263] to enhance our molecule processing model.

Redesigning Force Field. Our research has underscored the impact of FF parameterization and functional forms on overall performance. Developing functions that adhere more closely to the physics rules to fit potential energy is indeed a worthwhile research.

Enhancing Training Databases. Since our GB-FFs models belong to the data-driven approach, the quality and completeness of training data are important. We plan to collect more accurate training databases to further enhance model performance.

4.5 Supplementary Results

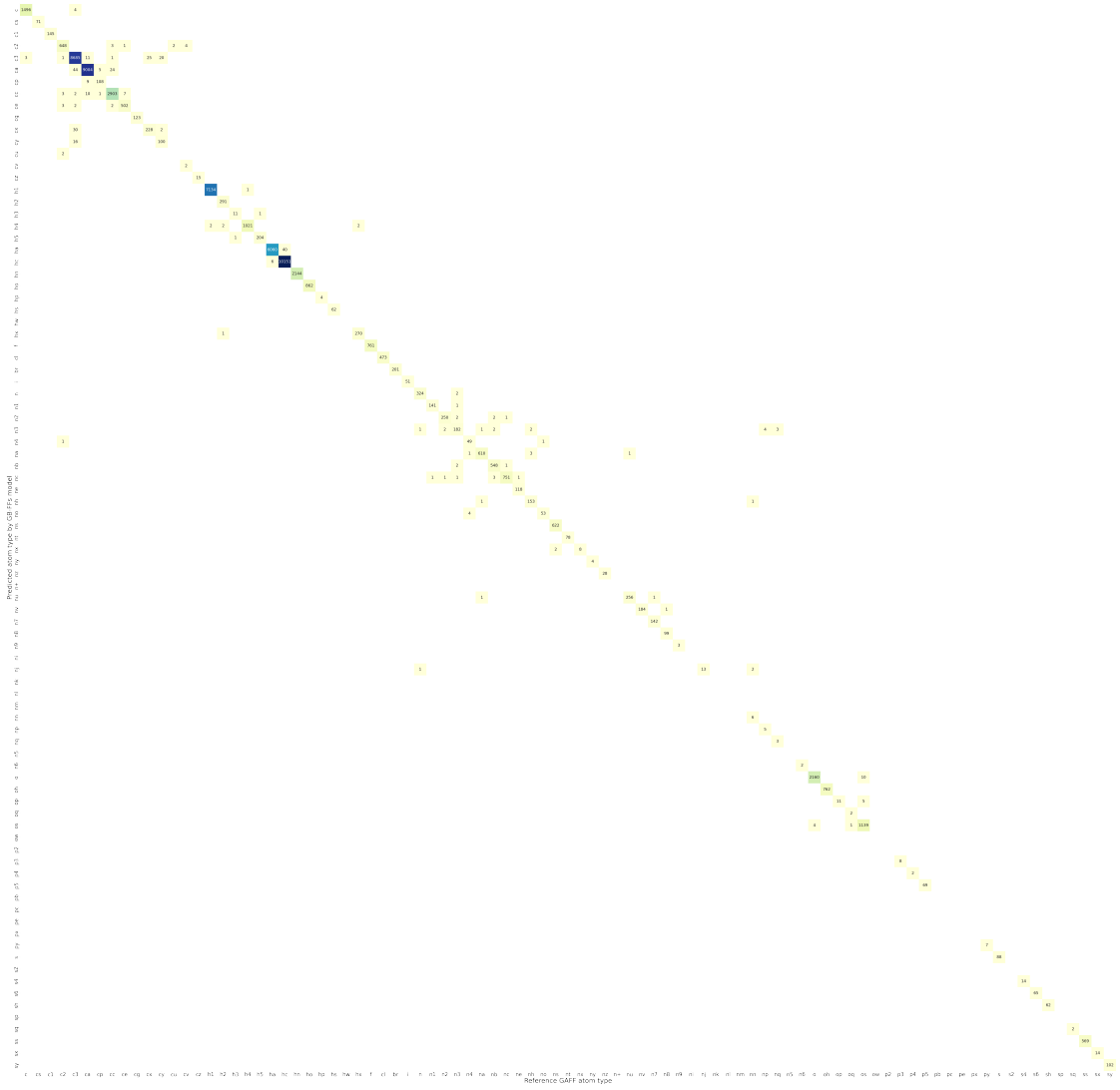

Figure 4.15: **Full results of predicted atom types on SPICE database.** The numbers in the table represent the frequency for each case.

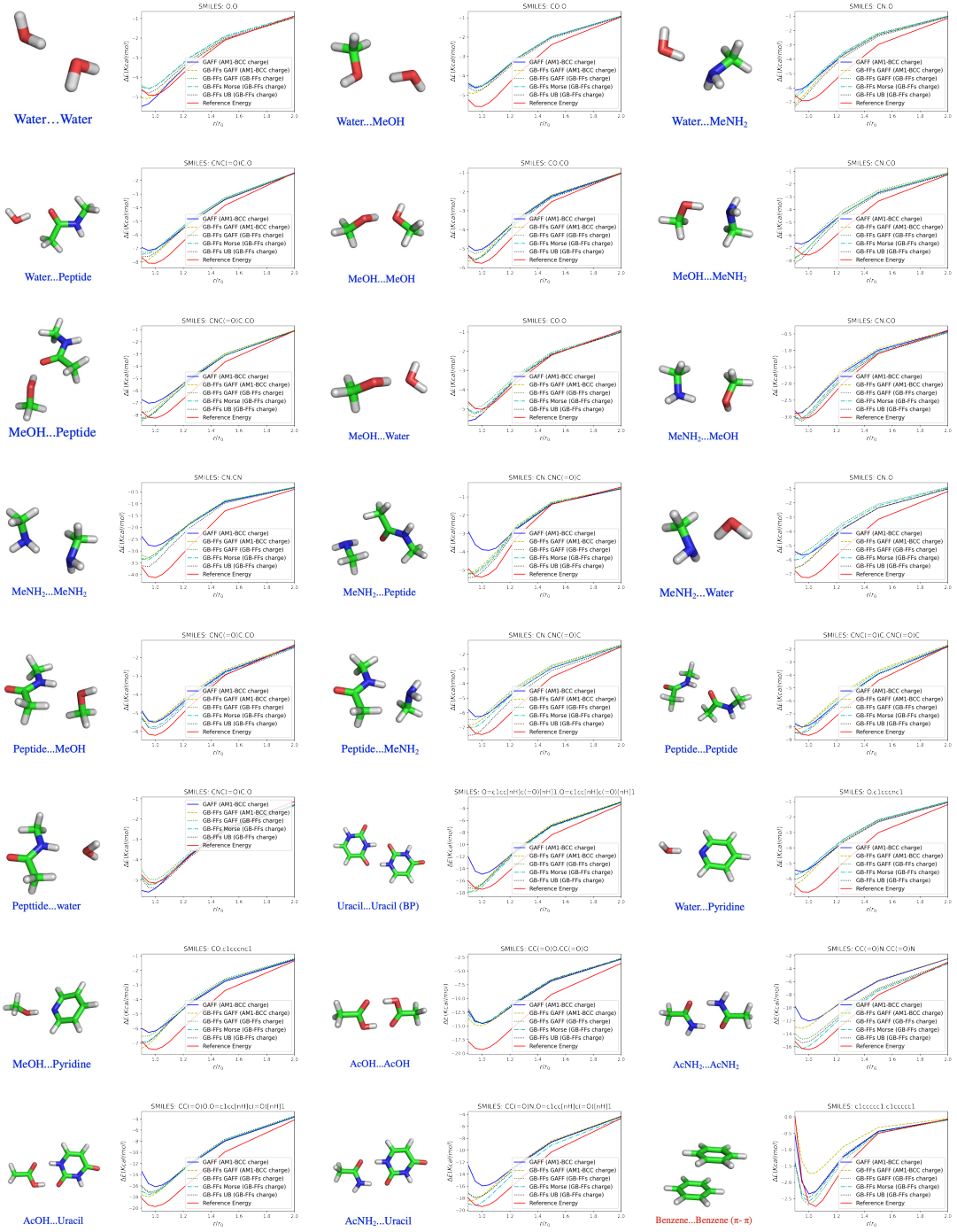

Figure 4.16: **All results for S66 × 8 database (1 / 3)**.

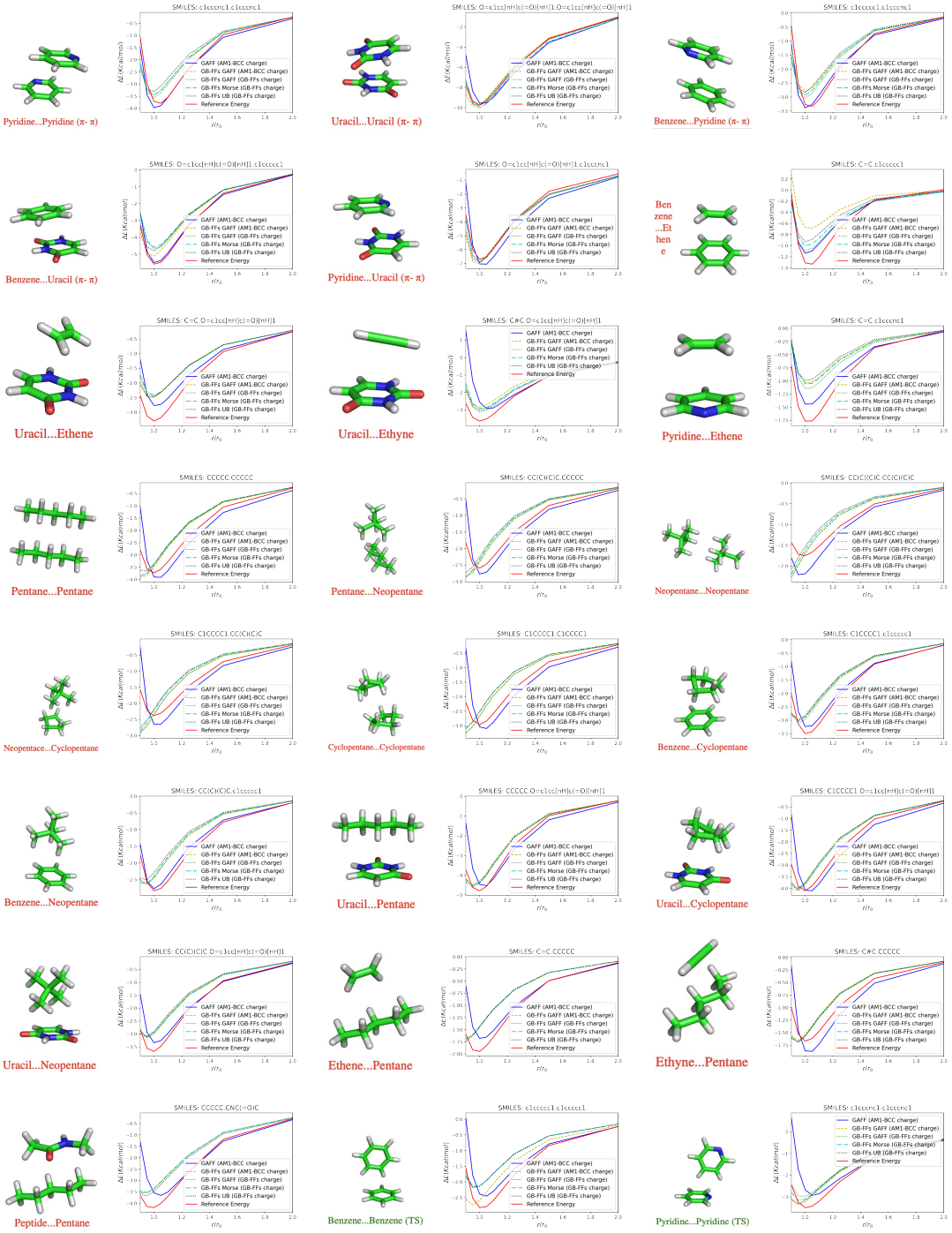

Figure 4.16: **All results for S66 × 8 database (2 / 3).**

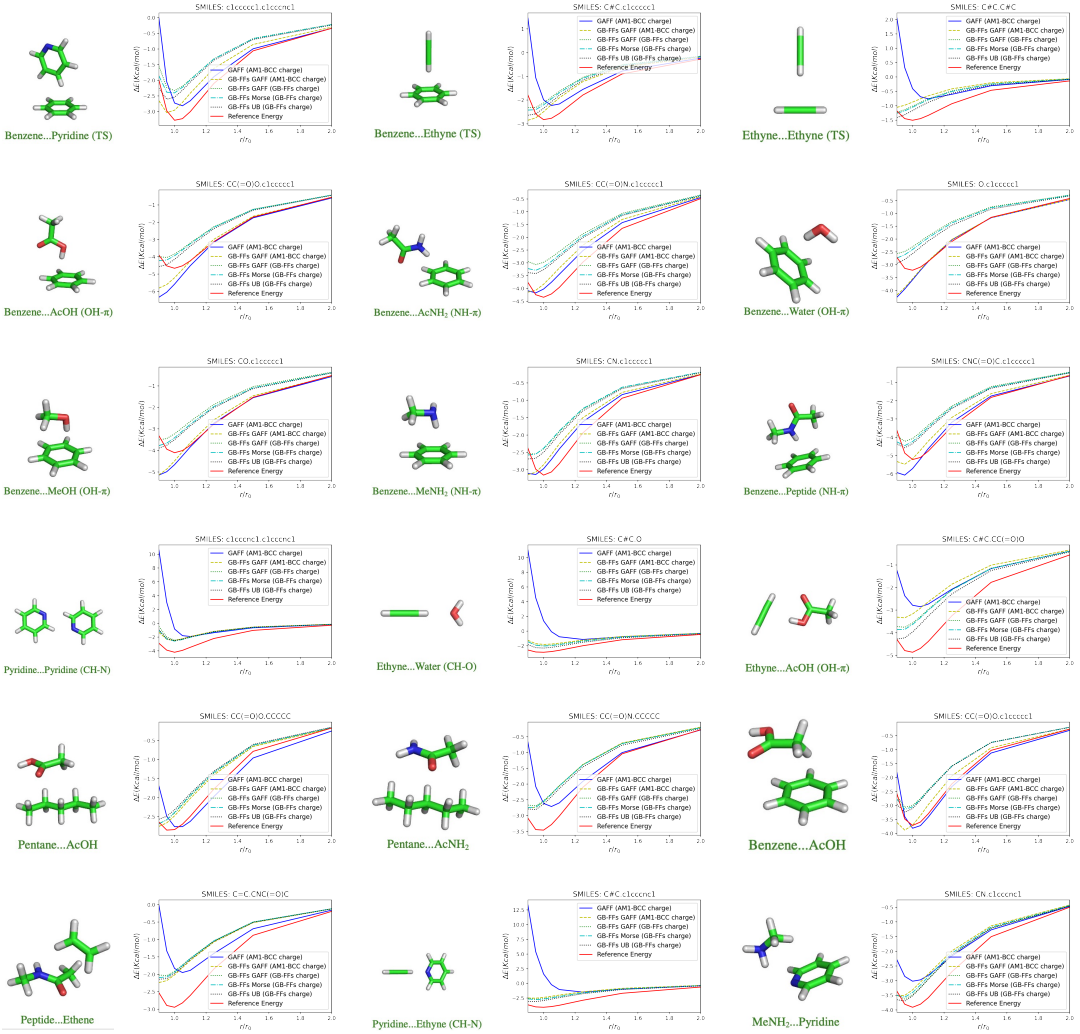

Figure 4.16: **All results for S66 × 8 database (3 / 3)**. The full results of "GAFF(AM1-BCC charge)", "GB-FFs GAFF (AM1-BCC charge)", "GB-FFs GAFF (GB-FFs charge)", "GB-FFs Morse (GB-FFs charge)" and "GB-FFs UB (GB-FFs charge)" on S66×8 database. For each dimer, the left subfigure is the representation (directly from paper [254], grey atom is hydrogen, red atom is carbon, blue atom is N, red atom is oxygen) and the right subfigure shows change of potential energy with the distance between two monomers (title is the SMILES of dimers, x-axis represents the ratio of inter-molecule distance to its equilibrium value (from 0.9 to 2.0), y-axis represents the relative potential energy (Kcal/mol)).

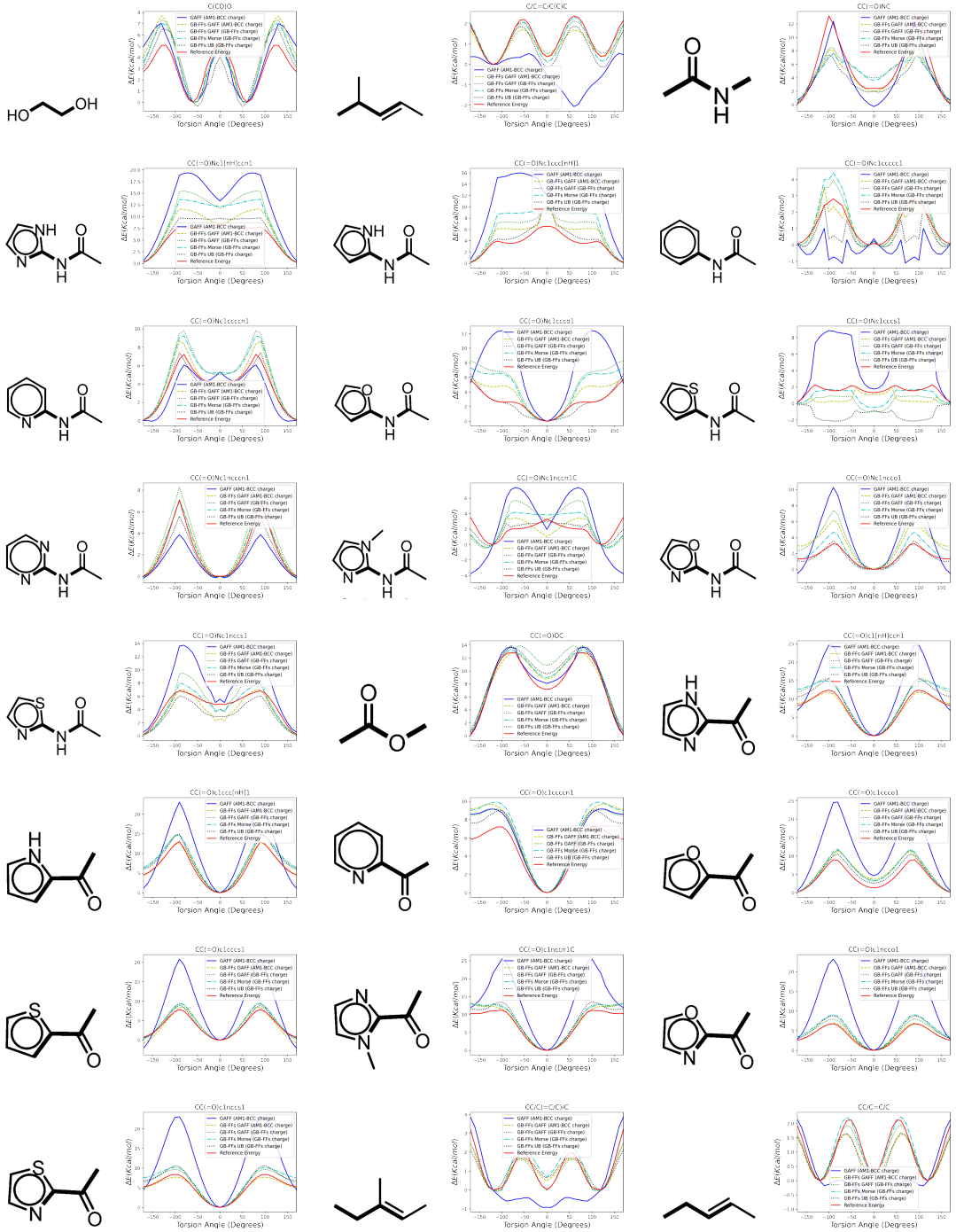

Figure 4.17: **All results for Torsion Scan database (1 / 3)**.

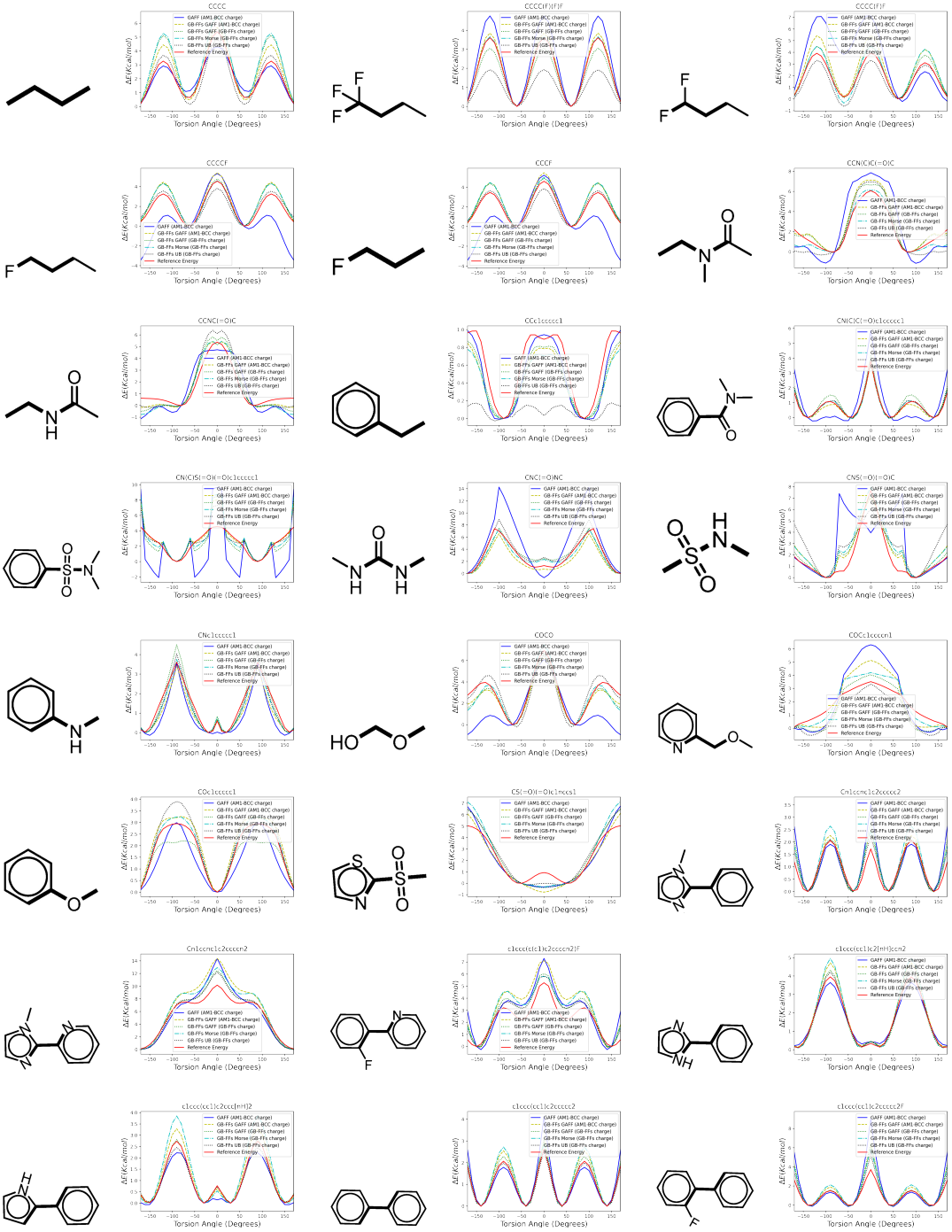

Figure 4.17: **All results for Torsion Scan database (2 / 3)**.

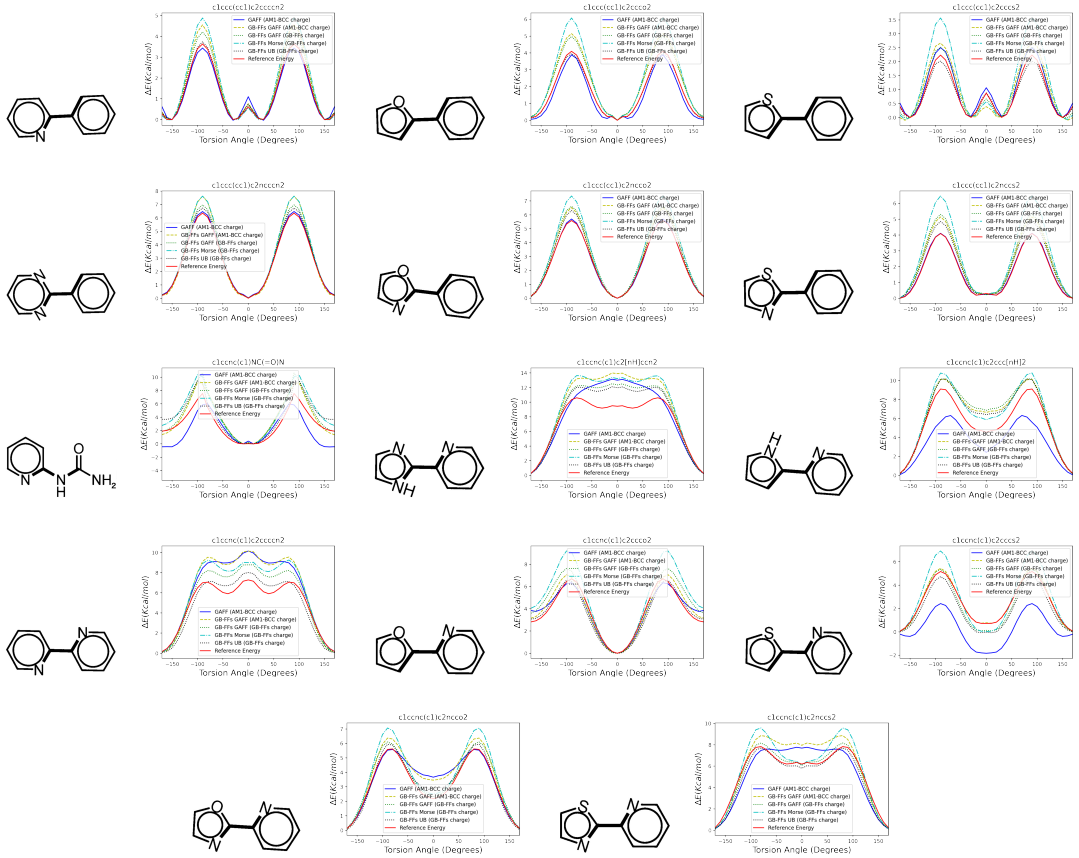

Figure 4.17: **All results for Torsion Scan database (3 / 3)**. The full results of "GAFF(AM1-BCC charge)", "GB-FFs GAFF (AM1-BCC charge)", "GB-FFs GAFF (GB-FFs charge)", "GB-FFs Morse (GB-FFs charge)" and "GB-FFs UB (GB-FFs charge)" on torsion scan database. For each drug-like fragment, the left subfigure is the representation (directly from paper [258], the bolded dihedral angle represents the varying dihedral angle) and the right subfigure shows change of potential energy with the dihedral angles (title is the SMILES of molecules, x-axis represents the degrees of dihedral angles (from -170° to 170°), y-axis represents the relative potential energy (Kcal/mol)).

Bibliography

[1] D. A. Pearlman 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". In: *Computer Physics Communications* 91.1-3 (1995), pp. 1–41.

[2] D. A. Case et al. "The Amber biomolecular simulation programs". In: *Journal of computational chemistry* 26.16 (2005), pp. 1668–1688.

[3] L. Yang et al. "New-generation amber united-atom force field". In: *The journal of physical chemistry B* 110.26 (2006), pp. 13166–13176.

[4] J. W. Ponder et al. "TINKER: Software tools for molecular design". In: *Washington University School of Medicine, Saint Louis, MO* 3 (2004).

[5] J. A. Rackers et al. "Tinker 8: software tools for molecular design". In: *Journal of chemical theory and computation* 14.10 (2018), pp. 5273–5289.

[6] *Tox21 Challenge*. `http://tripod.nih.gov/tox21/challenge/`.

[7] J. S. Smith, O. Isayev, and A. E. Roitberg. "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost". In: *Chemical science* 8.4 (2017), pp. 3192–3203.

[8] J. S. Smith, O. Isayev, and A. E. Roitberg. "ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules". In: *Scientific data* 4.1 (2017), pp. 1–8.

[9] T. Fink, H. Bruggesser, and J.-L. Reymond. "Virtual exploration of the small-molecule chemical universe below 160 daltons". In: *Angewandte Chemie International Edition* 44.10 (2005), pp. 1504–1508.

[10] T. Fink and J.-L. Reymond. "Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery". In: *Journal of chemical information and modeling* 47.2 (2007), pp. 342–353.

[11] J.-D. Chai and M. Head-Gordon. "Systematic optimization of long-range corrected hybrid density functionals". In: *The Journal of chemical physics* 128.8 (2008), p. 084106.

[12] R. Ditchfield, W. J. Hehre, and J. A. Pople. "Self-consistent molecular-orbital methods. IX. An extended Gaussian-type basis for molecular-orbital studies of organic molecules". In: *The Journal of Chemical Physics* 54.2 (1971), pp. 724–728.

[13] L. Daniel. *Home to the Chemical Reaction Database*. `https://kmt.vander-lingen.nl/`.

[14] P. Eastman et al. “SPICE, A Dataset of Drug-like Molecules and Peptides for Training Machine Learning Potentials”. In: *Scientific Data* 10.1 (2023), p. 11.

[15] A. Najibi and L. Goerigk. “The nonlocal kernel in van der Waals density functionals as an additive correction: An extensive analysis with special emphasis on the B97M-V and ωB97M-V approaches”. In: *Journal of Chemical Theory and Computation* 14.11 (2018), pp. 5725–5738.

[16] N. Mardirossian and M. Head-Gordon. “ω B97M-V: A combinatorially optimized, range-separated hybrid, meta-GGA density functional with VV10 nonlocal correlation”. In: *The Journal of chemical physics* 144.21 (2016), p. 214110.

[17] F. Weigend and R. Ahlrichs. “Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy”. In: *Physical Chemistry Chemical Physics* 7.18 (2005), pp. 3297–3305.

[18] D. Rappoport and F. Furche. “Property-optimized Gaussian basis sets for molecular response calculations”. In: *The Journal of chemical physics* 133.13 (2010), p. 134105.

[19] S. Kim et al. “PubChem 2019 update: improved access to chemical data”. In: *Nucleic acids research* 47.D1 (2019), pp. D1102–D1109.

[20] A. G. Donchev et al. “Quantum chemical benchmark databases of gold-standard dimer interaction energies”. In: *Scientific data* 8.1 (2021), p. 55.

[21] J. Behler. “Four generations of high-dimensional neural network potentials”. In: *Chemical Reviews* 121.16 (2021), pp. 10037–10072.

[22] O. T. Unke et al. “Machine learning force fields”. In: *Chemical Reviews* 121.16 (2021), pp. 10142–10186.

[23] E. Kocer, T. W. Ko, and J. Behler. “Neural network potentials: A concise overview of methods”. In: *Annual review of physical chemistry* 73 (2022), pp. 163–186.

[24] S. Batzner et al. “E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials”. In: *Nature communications* 13.1 (2022), p. 2453.

[25] T. J. Inizan et al. “Scalable hybrid deep neural networks/polarizable potentials biomolecular simulations including long-range effects”. In: *Chemical Science* (2023).

[26] Y. Li et al. “Machine learning force field parameters from ab initio data”. In: *Journal of chemical theory and computation* 13.9 (2017), pp. 4492–4503.

[27] R. Galvelis, S. Doerr, J. M. Damas, M. J. Harvey, and G. De Fabritiis. “A scalable molecular force field parameterization method based on density functional theory and quantum-level machine learning”. In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3485–3493.

[28] Y. Wang et al. “End-to-end differentiable construction of molecular mechanics force fields”. In: *Chemical Science* 13.41 (2022), pp. 12016–12033.

[29] C. Isert, K. Atz, and G. Schneider. “Structure-based drug design with geometric deep learning”. In: *Current Opinion in Structural Biology* 79 (2023), p. 102548.

[30] H. E. Sauceda, S. Chmiela, I. Poltavsky, K.-R. Müller, and A. Tkatchenko. “Molecular force fields with gradient-domain machine learning: Construction and application to dynamics of small molecules with coupled cluster forces”. In: *The Journal of chemical physics* 150.11 (2019), p. 114102.

[31] I. A. Guedes et al. “New machine learning and physics-based scoring functions for drug discovery”. In: *Scientific reports* 11.1 (2021), p. 3198.

[32] Q. Bai et al. "Application advances of deep learning methods for de novo drug design and molecular dynamics simulation". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 12.3 (2022), e1581.

[33] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[34] Y. Rong et al. "Self-supervised graph transformer on large-scale molecular data". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12559–12571.

[35] X. Fang et al. "Geometry-enhanced molecular representation learning for property prediction". In: *Nature Machine Intelligence* 4.2 (2022), pp. 127–134.

[36] J. Gasteiger, J. Groß, and S. Günnemann. "Directional message passing for molecular graphs". In: *arXiv preprint arXiv:2003.03123* (2020).

[37] Z. Yang, M. Chakraborty, and A. D. White. "Predicting chemical shifts with graph neural networks". In: *Chemical science* 12.32 (2021), pp. 10802–10809.

[38] L. Luo et al. "An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition". In: *Bioinformatics* 34.8 (2018), pp. 1381–1388.

[39] U. Naseem, K. Musial, P. Eklund, and M. Prasad. "Biomedical named-entity recognition by hierarchically fusing biobert representations and deep contextual-level word-embedding". In: *2020 International joint conference on neural networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[40] P. Schwaller et al. "Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy". In: *Chemical science* 11.12 (2020), pp. 3316–3325.

[41] P. Schwaller, A. C. Vaucher, T. Laino, and J.-L. Reymond. "Prediction of chemical reaction yields using deep learning". In: *Machine learning: science and technology* 2.1 (2021), p. 015016.

[42] P. Schwaller et al. "Mapping the space of chemical reactions using attention-based neural networks". In: *Nature machine intelligence* 3.2 (2021), pp. 144–152.

[43] S. Honda, S. Shi, and H. R. Ueda. "Smiles transformer: Pre-trained molecular fingerprint for low data drug discovery". In: *arXiv preprint arXiv:1911.04738* (2019).

[44] C. Li, J. Feng, S. Liu, and J. Yao. "A novel molecular representation learning for molecular property prediction with a multiple SMILES-based augmentation". In: *Computational Intelligence and Neuroscience* 2022 (2022).

[45] A. Yüksel, E. Ulusoy, A. Ünlü, and T. Doğan. "SELFormer: Molecular Representation Learning via SELFIES Language Models". In: *Machine Learning: Science and Technology* (2023).

[46] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013).

[47] M. Henaff, J. Bruna, and Y. LeCun. "Deep convolutional networks on graph-structured data". In: *arXiv preprint arXiv:1506.05163* (2015).

[48] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[49] A. Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[51] H. Dai, B. Dai, and L. Song. "Discriminative embeddings of latent variable models for structured data". In: *International conference on machine learning*. PMLR. 2016, pp. 2702–2711.

[52] K. Yang et al. "Analyzing learned molecular representations for property prediction". In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388.

[53] C. Qian, Y. Xiong, and X. Chen. "Directed graph attention neural network utilizing 3D coordinates for molecular property prediction". In: *Computational Materials Science* 200 (2021), p. 110761.

[54] X. Han, M. Jia, Y. Chang, Y. Li, and S. Wu. "Directed message passing neural network (D-MPNN) with graph edge attention (GEA) for property prediction of biofuel-relevant species". In: *Energy and AI* 10 (2022), p. 100201.

[55] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[56] O. Vinyals, S. Bengio, and M. Kudlur. "Order matters: Sequence to sequence for sets". In: *arXiv preprint arXiv:1511.06391* (2015).

[57] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[58] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. "Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs". In: *arXiv preprint arXiv:1811.01900* (2018).

[59] A. K. Rappe and W. A. Goddard III. "Charge equilibration for molecular dynamics simulations". In: *The Journal of Physical Chemistry* 95.8 (1991), pp. 3358–3363.

[60] Y. Wang et al. "EspalomaCharge: Machine learning-enabled ultra-fast partial charge assignment". In: *arXiv preprint arXiv:2302.06758* (2023).

[61] A. Grisafi, D. M. Wilkins, G. Csányi, and M. Ceriotti. "Symmetry-adapted machine learning for tensorial properties of atomistic systems". In: *Physical review letters* 120.3 (2018), p. 036002.

[62] P. Gkeka et al. "Machine learning force fields and coarse-grained variables in molecular dynamics: application to materials and biological systems". In: *Journal of chemical theory and computation* 16.8 (2020), pp. 4757–4775.

[63] F. Musil, A. Grisafi, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti. "Physics-inspired structural representations for molecules and materials". In: *Chemical Reviews* 121.16 (2021), pp. 9759–9815.

[64] I. Chahrour and J. Wells. "Comparing machine learning and interpolation methods for loop-level calculations". In: *SciPost Physics* 12.6 (2022), p. 187.

[65] Y. Chen, B. Dong, and J. Xu. "Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations". In: *Journal of computational physics* 455 (2022), p. 110996.

[66] P. Grigorev, A. M. Goryaeva, M.-C. Marinica, J. R. Kermode, and T. D. Swinburne. "Calculation of dislocation binding to helium-vacancy defects in tungsten using hybrid ab initio-machine learning methods". In: *Acta Materialia* 247 (2023), p. 118734.

[67] K. Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[69] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[70] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. “Is object localization for free?-weakly-supervised learning with convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 685–694.

[71] S. Sudholt and G. A. Fink. “Phocnet: A deep convolutional neural network for word spotting in handwritten documents”. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE. 2016, pp. 277–282.

[72] D. Weininger. “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules”. In: *Journal of chemical information and computer sciences* 28.1 (1988), pp. 31–36.

[73] G. Chen and Y. Maday. “Directed message passing based on attention for prediction of molecular properties”. In: *Computational Materials Science* 229 (2023), p. 112443.

[74] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. “Extensions of marginalized graph kernels”. In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 70.

[75] G. Landrum. *RDKit: Open-source cheminformatics*. `https://www.rdkit.org`.

[76] P. Muller. “Glossary of terms used in physical organic chemistry (IUPAC Recommendations 1994)”. In: *Pure and Applied Chemistry* 66.5 (1994), pp. 1077–1184.

[77] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[78] Z. Wu et al. “MoleculeNet: a benchmark for molecular machine learning”. In: *Chemical science* 9.2 (2018), pp. 513–530.

[79] U. Burkert. “N. L. Allinger, Molecular Mechanics”. In: *ACS, Washington, DC* (1982).

[80] V. Poltev. *Molecular mechanics: principles, history, and current status*. 2015.

[81] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case. “Development and testing of a general amber force field”. In: *Journal of computational chemistry* 25.9 (2004), pp. 1157–1174.

[82] J. E. Jones. “On the determination of molecular fields.—I. From the variation of the viscosity of a gas with temperature”. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106.738 (1924), pp. 441–462.

[83] J. E. Jones. “On the determination of molecular fields.—II. From the equation of state of a gas”. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106.738 (1924), pp. 463–477.

[84] C. I. Bayly, P. Cieplak, W. Cornell, and P. A. Kollman. “A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: the RESP model”. In: *The Journal of Physical Chemistry* 97.40 (1993), pp. 10269–10280.

[85] W. D. Cornell, P. Cieplak, C. I. Bayly, and P. A. Kollman. "Application of RESP charges to calculate conformational energies, hydrogen bond energies, and free energies of solvation". In: *Journal of the American Chemical Society* 115.21 (2002), pp. 9620–9631.

[86] H. A. Lorentz. "Ueber die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase". In: *Annalen der physik* 248.1 (1881), pp. 127–136.

[87] D. Berthelot. "Sur le mélange des gaz". In: *Compt. Rendus* 126.3 (1898), p. 15.

[88] L.-P. Wang, J. Chen, and T. Van Voorhis. "Systematic parametrization of polarizable force fields from quantum chemistry data". In: *Journal of chemical theory and computation* 9.1 (2013), pp. 452–460.

[89] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "SMU: smooth activation function for deep networks using smoothing maximum technique". In: *arXiv preprint arXiv:2111.04682* (2021).

[90] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. "Maxout networks". In: *International conference on machine learning*. PMLR. 2013, pp. 1319–1327.

[91] J. P. Devlin. "Urey-Bradley"Nonbonded"Forces". In: *The Journal of Chemical Physics* 39.9 (1963), pp. 2385–2385.

[92] J. Wang, W. Wang, P. A. Kollman, and D. A. Case. "Automatic atom type and bond type perception in molecular mechanical calculations". In: *Journal of molecular graphics and modelling* 25.2 (2006), pp. 247–260.

[93] I. Batatia, D. P. Kovacs, G. Simm, C. Ortner, and G. Csányi. "MACE: Higher order equivariant message passing neural networks for fast and accurate force fields". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 11423–11436.

[94] Z. Liu, Y. Yang, and Q. Cai. "Neural network as a function approximator and its application in solving differential equations". In: *Applied Mathematics and Mechanics* 40.2 (2019), pp. 237–248.

[95] P. Tchébychev. *Sur les questions de minima qui se rattechent a la raprésentation aproximative des fonctions*. Imprimerie de l'Academie Impériale des Sciences, 1858.

[96] J. L. de Lagrange. "Leçons élémentaires sur les Mathématiques, données à l'École normale, en 1795". In: *Oeuvres de Lagrange* 7 (1812), pp. 183–287.

[97] M. Abramowitz, I. A. Stegun, and R. H. Romer. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. 1988.

[98] L. Andrews. "Special functions for engineers and applied mathematicians". In: *Applied Optics* 25.18 (1986), p. 3096.

[99] K. Banerjee. *Generalized inverse of matrices and its applications*. 1973.

[100] S. L. Campbell and C. D. Meyer. *Generalized inverses of linear transformations*. SIAM, 2009.

[101] F. W. Luttmann and T. J. Rivlin. "Some numerical experiments in the theory of polynomial interpolation". In: *IBM Journal of Research and Development* 9.3 (1965), pp. 187–191.

[102] G. M. Phillips. *Interpolation and approximation by polynomials*. Vol. 14. Springer Science & Business Media, 2003.

[103] B. A. Ibrahimoglu. "Lebesgue functions and Lebesgue constants in polynomial interpolation". In: *Journal of Inequalities and Applications* 2016 (2016), pp. 1–15.

[104] C. Runge. "Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten". In: *Zeitschrift für Mathematik und Physik* 46.224-243 (1901), p. 20.

[105] A. Schönhage. "Fehlerfortpflanzung bei interpolation". In: *Numerische Mathematik* 3.1 (1961), pp. 62–71.

[106] A. Turetskii. "The bounding of polynomials prescribed at equally distributed points". In: *Proc. Pedag. Inst. Vitebsk.* Vol. 3. 1940, pp. 117–127.

[107] R. Günttner. "Evaluation of Lebesgue constants". In: *SIAM Journal on Numerical Analysis* 17.4 (1980), pp. 512–520.

[108] H. Ehlich and K. Zeller. "Auswertung der normen von interpolationsoperatoren". In: *Mathematische Annalen* 164.2 (1966), pp. 105–112.

[109] J. H. McCabe and G. M. Phillips. "On a certain class of Lebesgue constants". In: *BIT Numerical Mathematics* 13.4 (1973), pp. 434–442.

[110] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

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

[112] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[113] M. He, Z. Wei, and J.-R. Wen. "Convolutional neural networks on graphs with chebyshev approximation, revisited". In: *arXiv preprint arXiv:2202.03580* (2022).

[114] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. "Inference of regular expressions for text extraction from examples". In: *IEEE Transactions on Knowledge and Data Engineering* 28.5 (2016), pp. 1217–1230.

[115] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan. "The first question generation shared task evaluation challenge". In: (2010).

[116] J. Zhang, C. Zong, et al. "Deep Neural Networks in Machine Translation: An Overview." In: *IEEE Intell. Syst.* 30.5 (2015), pp. 16–25.

[117] K. Kukich. "Techniques for automatically correcting words in text". In: *Acm Computing Surveys (CSUR)* 24.4 (1992), pp. 377–439.

[118] G. Neglur, R. L. Grossman, and B. Liu. "Assigning unique keys to chemical compounds for data integration: Some interesting counter examples". In: *International workshop on data integration in the life sciences*. Springer. 2005, pp. 145–157.

[119] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[120] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[121] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[122] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[123] K. Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[124] M.-T. Luong, H. Pham, and C. D. Manning. “Effective approaches to attention-based neural machine translation”. In: *arXiv preprint arXiv:1508.04025* (2015).

[125] Y. Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[126] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the limits of language modeling”. In: *arXiv preprint arXiv:1602.02410* (2016).

[127] Y. Kim, C. Denton, L. Hoang, and A. M. Rush. “Structured attention networks”. In: *arXiv preprint arXiv:1702.00887* (2017).

[128] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[129] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[130] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[131] A. Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[132] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. “ZINC: a free tool to discover chemistry for biology”. In: *Journal of chemical information and modeling* 52.7 (2012), pp. 1757–1768.

[133] J. J. Irwin and B. K. Shoichet. “ZINC- a free database of commercially available compounds for virtual screening”. In: *Journal of chemical information and modeling* 45.1 (2005), pp. 177–182.

[134] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. O. Falcao. “A Bayesian approach to in silico blood-brain barrier penetration modeling”. In: *Journal of chemical information and modeling* 52.6 (2012), pp. 1686–1697.

[135] M. Kuhn, M. Campillos, I. Letunic, L. J. Jensen, and P. Bork. “A side effect resource to capture phenotypic effects of drugs”. In: *Molecular systems biology* 6.1 (2010), p. 343.

[136] M. Kuhn, I. Letunic, L. J. Jensen, and P. Bork. “The SIDER database of drugs and side effects”. In: *Nucleic acids research* 44.D1 (2016), pp. D1075–D1079.

[137] *Medical Dictionary for Regulatory Activities*. `http://www.meddra.org/`.

[138] *Exploring ToxCast Data*. `https://www.epa.gov/chemical-research/exploring-toxcast-data`.

[139] G. Subramanian, B. Ramsundar, V. Pande, and R. A. Denny. “Computational modeling of β-secretase 1 (BACE-1) inhibitors using ligand based approaches”. In: *Journal of chemical information and modeling* 56.10 (2016), pp. 1936–1949.

[140] *AIDS Antiviral Screen Data*. `http://wiki.nci.nih.gov/display/`.

[141] S. G. Rohrer and K. Baumann. “Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data”. In: *Journal of chemical information and modeling* 49.2 (2009), pp. 169–184.

[142] *National Library of Medicine*. `https://pubchem.ncbi.nlm.nih.gov/docs/bioassays`.

[143] J. S. Delaney. "ESOL: estimating aqueous solubility directly from molecular structure". In: *Journal of chemical information and computer sciences* 44.3 (2004), pp. 1000–1005.

[144] D. L. Mobley and J. P. Guthrie. "FreeSolv: a database of experimental and calculated hydration free energies, with input files". In: *Journal of computer-aided molecular design* 28 (2014), pp. 711–720.

[145] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld. "Fast and accurate modeling of molecular atomization energies with machine learning". In: *Physical review letters* 108.5 (2012), p. 058301.

[146] L. C. Blum and J.-L. Reymond. "970 million druglike small molecules for virtual screening in the chemical universe database GDB-13". In: *Journal of the American Chemical Society* 131.25 (2009), pp. 8732–8733.

[147] R. Ramakrishnan, M. Hartmann, E. Tapavicza, and O. A. Von Lilienfeld. "Electronic spectra from TDDFT and machine learning in chemical space". In: *The Journal of chemical physics* 143.8 (2015), p. 084111.

[148] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond. "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17". In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2864–2875.

[149] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific data* 1.1 (2014), pp. 1–7.

[150] G. W. Bemis and M. A. Murcko. "The properties of known drugs. 1. Molecular frameworks". In: *Journal of medicinal chemistry* 39.15 (1996), pp. 2887–2893.

[151] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[152] M. E. Peters et al. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365v2* (2018).

[153] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. "Improving language understanding with unsupervised learning". In: *OpenAI* (2018).

[154] A. M. Dai and Q. V. Le. "Semi-supervised sequence learning". In: *Advances in neural information processing systems* 28 (2015), pp. 3079–3087.

[155] J. Howard and S. Ruder. "Universal language model fine-tuning for text classification". In: *arXiv preprint arXiv:1801.06146* (2018).

[156] A. Williams, N. Nangia, and S. R. Bowman. "A broad-coverage challenge corpus for sentence understanding through inference". In: *arXiv preprint arXiv:1704.05426* (2017).

[157] E. F. Sang and F. De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *arXiv preprint cs/0306050* (2003).

[158] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

[159] W. L. Taylor. ""Cloze procedure": A new tool for measuring readability". In: *Journalism quarterly* 30.4 (1953), pp. 415–433.

[160] Y. Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[161] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang. "SMILES-BERT: large scale unsupervised pre-training for molecular property prediction". In: *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*. 2019, pp. 429–436.

[162] W. Hu et al. "Strategies for pre-training graph neural networks". In: *arXiv preprint arXiv:1905.12265* (2019).

[163] R. Irwin, S. Dimitriadis, J. He, and E. J. Bjerrum. "Chemformer: A Pre-Trained Transformer for Computational Chemistry". In: *Machine Learning: Science and Technology* (2021).

[164] J. Payne, M. Srouji, D. A. Yap, and V. Kosaraju. "BERT Learns (and Teaches) Chemistry". In: *arXiv preprint arXiv:2007.16012* (2020).

[165] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[166] Z. Xiong et al. "Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism". In: *Journal of medicinal chemistry* 63.16 (2019), pp. 8749–8760.

[167] A. Sperduti and A. Starita. "Supervised neural networks for the classification of structures". In: *IEEE Transactions on Neural Networks* 8.3 (1997), pp. 714–735.

[168] M. Gori, G. Monfardini, and F. Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE international joint conference on neural networks*. Vol. 2. 2005. 2005, pp. 729–734.

[169] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[170] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[171] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. "Graphrnn: Generating realistic graphs with deep auto-regressive models". In: *International conference on machine learning*. PMLR. 2018, pp. 5708–5717.

[172] H. Wang et al. "Graphgan: Graph representation learning with generative adversarial nets". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 2018.

[173] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec. "Graph convolutional policy network for goal-directed molecular graph generation". In: *Advances in neural information processing systems* 31 (2018).

[174] W. Jin, R. Barzilay, and T. Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *International conference on machine learning*. PMLR. 2018, pp. 2323–2332.

[175] W. W. Zachary. "An information flow model for conflict and fission in small groups". In: *Journal of anthropological research* 33.4 (1977), pp. 452–473.

[176] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[177] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[178] H. Liu, H. Kou, C. Yan, and L. Qi. "Link prediction in paper citation network to construct paper correlation graph". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–12.

[179] D. K. Duvenaud et al. "Convolutional networks on graphs for learning molecular fingerprints". In: *Advances in neural information processing systems* 28 (2015).

[180] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley. "Molecular graph convolutions: moving beyond fingerprints". In: *Journal of computer-aided molecular design* 30.8 (2016), pp. 595–608.

[181] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. "Generative adversarial text to image synthesis". In: *International conference on machine learning*. PMLR. 2016, pp. 1060–1069.

[182] T. Xu et al. "Attngan: Fine-grained text to image generation with attentional generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1316–1324.

[183] T. Qiao, J. Zhang, D. Xu, and D. Tao. "Mirrorgan: Learning text-to-image generation by redescription". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1505–1514.

[184] A. Allera, A. M. Goryaeva, P. Lafourcade, J.-B. Maillet, and M.-C. Marinica. "Neighbors Map: an Efficient Atomic Descriptor for Structural Analysis". In: *arXiv preprint arXiv:2307.00978* (2023).

[185] M. Zhang and Y. Chen. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[186] Z. Wang, Z. Ren, C. He, P. Zhang, and Y. Hu. "Robust Embedding with Multi-Level Structures for Link Prediction." In: *IJCAI*. 2019, pp. 5240–5246.

[187] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[188] D. Bo, X. Wang, C. Shi, and H. Shen. "Beyond low-frequency information in graph convolutional networks". In: *arXiv preprint arXiv:2101.00797* (2021).

[189] P. B. Jørgensen, K. W. Jacobsen, and M. N. Schmidt. "Neural message passing with edge updates for predicting properties of molecules and materials". In: *arXiv preprint arXiv:1806.03146* (2018).

[190] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 3438–3445.

[191] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[192] K. Ishiguro, S.-i. Maeda, and M. Koyama. "Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis". In: *arXiv preprint arXiv:1902.01020* (2019).

[193] D. L. Mobley et al. "Escaping atom types in force fields using direct chemical perception". In: *Journal of chemical theory and computation* 14.11 (2018), pp. 6076–6092.

[194] D. A. Case et al. *Amber 10*. Tech. rep. University of California, 2008.

[195] A. Messiah. *Quantum mechanics*. Courier Corporation, 2014.

[196] M. Born and W. Heisenberg. "Zur quantentheorie der molekeln". In: *Original Scientific Papers Wissenschaftliche Originalarbeiten* (1985), pp. 216–246.

[197] L. Pauling, R. B. Corey, and H. R. Branson. "The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain". In: *Proceedings of the National Academy of Sciences* 37.4 (1951), pp. 205–211.

[198] J. D. Watson and F. H. Crick. "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid". In: *Nature* 171.4356 (1953), pp. 737–738.

[199] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. "Comparison of simple potential functions for simulating liquid water". In: *The Journal of chemical physics* 79.2 (1983), pp. 926–935.

[200] M. W. Mahoney and W. L. Jorgensen. "A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions". In: *The Journal of chemical physics* 112.20 (2000), pp. 8910–8922.

[201] H. Berendsen, J. Grigera, and T. Straatsma. "The missing term in effective pair potentials". In: *Journal of Physical Chemistry* 91.24 (1987), pp. 6269–6271.

[202] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. a. Swaminathan, and M. Karplus. "CHARMM: a program for macromolecular energy, minimization, and dynamics calculations". In: *Journal of computational chemistry* 4.2 (1983), pp. 187–217.

[203] S. Patel and C. L. Brooks III. "CHARMM fluctuating charge force field for proteins: I parameterization and application to bulk organic liquid simulations". In: *Journal of computational chemistry* 25.1 (2004), pp. 1–16.

[204] V. M. Anisimov, G. Lamoureux, I. V. Vorobyov, N. Huang, B. Roux, and A. D. MacKerell. "Determination of electrostatic parameters for a polarizable force field based on the classical Drude oscillator". In: *Journal of chemical theory and computation* 1.1 (2005), pp. 153–168.

[205] B. R. Brooks et al. "CHARMM: the biomolecular simulation program". In: *Journal of computational chemistry* 30.10 (2009), pp. 1545–1614.

[206] F. Momany, R. F. McGuire, A. Burgess, and H. A. Scheraga. "Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids". In: *The Journal of Physical Chemistry* 79.22 (1975), pp. 2361–2381.

[207] Y. A. Arnautova, A. Jagielska, and H. A. Scheraga. "A new force field (ECEPP-05) for peptides, proteins, and organic molecules". In: *The Journal of Physical Chemistry B* 110.10 (2006), pp. 5025–5044.

[208] T. A. Halgren. "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94". In: *Journal of computational chemistry* 17.5-6 (1996), pp. 490–519.

[209] T. A. Halgren. "MMFF VII. Characterization of MMFF94, MMFF94s, and other widely available force fields for conformational energies and for intermolecular-interaction energies and geometries". In: *Journal of Computational Chemistry* 20.7 (1999), pp. 730–748.

[210] W. R. Scott et al. “The GROMOS biomolecular simulation program package”. In: *The Journal of Physical Chemistry A* 103.19 (1999), pp. 3596–3607.

[211] C. Oostenbrink, A. Villa, A. E. Mark, and W. F. Van Gunsteren. “A biomolecular force field based on the free enthalpy of hydration and solvation: the GROMOS force-field parameter sets 53A5 and 53A6”. In: *Journal of computational chemistry* 25.13 (2004), pp. 1656–1676.

[212] M. Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of computational chemistry* 26.16 (2005), pp. 1719–1751.

[213] W. L. Jorgensen and J. Tirado-Rives. “The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin”. In: *Journal of the American Chemical Society* 110.6 (1988), pp. 1657–1666.

[214] W. Damm, A. Frontera, J. Tirado–Rives, and W. L. Jorgensen. “OPLS all-atom force field for carbohydrates”. In: *Journal of computational chemistry* 18.16 (1997), pp. 1955–1970.

[215] P. Ren and J. W. Ponder. “Polarizable atomic multipole water model for molecular mechanics simulation”. In: *The Journal of Physical Chemistry B* 107.24 (2003), pp. 5933–5947.

[216] P. Ren and J. W. Ponder. “Temperature and pressure dependence of the AMOEBA water model”. In: *The Journal of Physical Chemistry B* 108.35 (2004), pp. 13427–13437.

[217] J. W. Ponder et al. “Current status of the AMOEBA polarizable force field”. In: *The journal of physical chemistry B* 114.8 (2010), pp. 2549–2564.

[218] P. Ren, C. Wu, and J. W. Ponder. “Polarizable atomic multipole-based molecular mechanics for organic molecules”. In: *Journal of chemical theory and computation* 7.10 (2011), pp. 3143–3161.

[219] C. Liu, J.-P. Piquemal, and P. Ren. “AMOEBA+ classical potential for modeling molecular interactions”. In: *Journal of chemical theory and computation* 15.7 (2019), pp. 4122–4139.

[220] J. A. Lemkul, J. Huang, B. Roux, and A. D. MacKerell Jr. “An empirical polarizable force field based on the classical drude oscillator model: development history and recent applications”. In: *Chemical reviews* 116.9 (2016), pp. 4983–5013.

[221] N. Gresh, G. A. Cisneros, T. A. Darden, and J.-P. Piquemal. “Anisotropic, polarizable molecular mechanics studies of inter-and intramolecular interactions and ligand- macromolecule complexes. A bottom-up strategy”. In: *Journal of chemical theory and computation* 3.6 (2007), pp. 1960–1986.

[222] S. Naseem-Khan et al. “Development of the Quantum-Inspired SIBFA Many-Body Polarizable Force Field: Enabling Condensed-Phase Molecular Dynamics Simulations”. In: *Journal of Chemical Theory and Computation* 18.6 (2022), pp. 3607–3621.

[223] T. Lelievre and G. Stoltz. “Partial differential equations and stochastic methods in molecular dynamics”. In: *Acta Numerica* 25 (2016), pp. 681–880.

[224] N. Raimbault, A. Grisafi, M. Ceriotti, and M. Rossi. “Using Gaussian process regression to simulate the vibrational Raman spectra of molecular crystals”. In: *New Journal of Physics* 21.10 (2019), p. 105001.

[225] G. M. Sommers, M. F. C. Andrade, L. Zhang, H. Wang, and R. Car. “Raman spectrum and polarizability of liquid water from deep neural networks”. In: *Physical Chemistry Chemical Physics* 22.19 (2020), pp. 10592–10602.

[226] N. Artrith and J. Behler. "High-dimensional neural network potentials for metal surfaces: A prototype study for copper". In: *Physical Review B* 85.4 (2012), p. 045439.

[227] A. M. Goryaeva, J.-B. Maillet, and M.-C. Marinica. "Towards better efficiency of interatomic linear machine learning potentials". In: *Computational Materials Science* 166 (2019), pp. 200–209.

[228] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren. "Neural network models of potential energy surfaces". In: *The Journal of chemical physics* 103.10 (1995), pp. 4129–4137.

[229] S. Lorenz, A. Groß, and M. Scheffler. "Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks". In: *Chemical Physics Letters* 395.4-6 (2004), pp. 210–215.

[230] J. Behler and M. Parrinello. "Generalized neural-network representation of high-dimensional potential-energy surfaces". In: *Physical review letters* 98.14 (2007), p. 146401.

[231] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. "Schnet–a deep learning architecture for molecules and materials". In: *The Journal of Chemical Physics* 148.24 (2018), p. 241722.

[232] O. T. Unke and M. Meuwly. "PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges". In: *Journal of chemical theory and computation* 15.6 (2019), pp. 3678–3693.

[233] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi. "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons". In: *Physical review letters* 104.13 (2010), p. 136403.

[234] S. Chmiela, H. E. Sauceda, I. Poltavsky, K.-R. Müller, and A. Tkatchenko. "sGDML: Constructing accurate and data efficient molecular force fields using machine learning". In: *Computer Physics Communications* 240 (2019), pp. 38–45.

[235] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker. "Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials". In: *Journal of Computational Physics* 285 (2015), pp. 316–330.

[236] M. A. Wood and A. P. Thompson. "Extending the accuracy of the SNAP interatomic potential form". In: *The Journal of chemical physics* 148.24 (2018), p. 241721.

[237] A. V. Shapeev. "Moment tensor potentials: A class of systematically improvable interatomic potentials". In: *Multiscale Modeling & Simulation* 14.3 (2016), pp. 1153–1173.

[238] R. Drautz. "Atomic cluster expansion for accurate and transferable interatomic potentials". In: *Physical Review B* 99.1 (2019), p. 014104.

[239] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong. "Graph networks as a universal machine learning framework for molecules and crystals". In: *Chemistry of Materials* 31.9 (2019), pp. 3564–3572.

[240] A. E. Allen, G. Dusson, C. Ortner, and G. Csányi. "Atomic permutationally invariant polynomials for fitting molecular force fields". In: *Machine Learning: Science and Technology* 2.2 (2021), p. 025017.

[241] R. M. Balabin and E. I. Lomakina. "Support vector machine regression (LS-SVM)—an alternative to artificial neural networks (ANNs) for the analysis of quantum chemistry data?" In: *Physical Chemistry Chemical Physics* 13.24 (2011), pp. 11710–11718.

[242] L. Lagardère et al. “Tinker-HP: a massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields”. In: *Chemical science* 9.4 (2018), pp. 956–972.

[243] O. Adjoua et al. “Tinker-HP: Accelerating molecular dynamics simulations of large complex systems with advanced point dipole polarizable force fields using GPUs and multi-GPU systems”. In: *Journal of chemical theory and computation* 17.4 (2021), pp. 2034–2053.

[244] J. C. Wu, G. Chattree, and P. Ren. “Automation of AMOEBA polarizable force field parameterization for small molecules”. In: *Theoretical chemistry accounts* 131 (2012), pp. 1–11.

[245] B. Walker, C. Liu, E. Wait, and P. Ren. “Automation of AMOEBA polarizable force field for small molecules: Poltype 2”. In: *Journal of Computational Chemistry* 43.23 (2022), pp. 1530–1542.

[246] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen. “GROMACS: fast, flexible, and free”. In: *Journal of computational chemistry* 26.16 (2005), pp. 1701–1718.

[247] A. Jakalian, B. L. Bush, D. B. Jack, and C. I. Bayly. “Fast, efficient generation of high-quality atomic charges. AM1-BCC model: I. Method”. In: *Journal of computational chemistry* 21.2 (2000), pp. 132–146.

[248] A. Jakalian, D. B. Jack, and C. I. Bayly. “Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and validation”. In: *Journal of computational chemistry* 23.16 (2002), pp. 1623–1641.

[249] S. Liu, H. Wang, W. Liu, J. Lasenby, H. Guo, and J. Tang. “Pre-training molecular graph representation with 3d geometry”. In: *arXiv preprint arXiv:2110.07728* (2021).

[250] M. K. Gilson, H. S. Gilson, and M. J. Potter. “Fast assignment of accurate partial atomic charges: an electronegativity equalization method that accounts for alternate resonance forms”. In: *Journal of chemical information and computer sciences* 43.6 (2003), pp. 1982–1997.

[251] P. M. Morse. “Diatomic molecules according to the wave mechanics. II. Vibrational levels”. In: *Physical review* 34.1 (1929), p. 57.

[252] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon. “A fifth-order perturbation comparison of electron correlation theories”. In: *Chemical Physics Letters* 157.6 (1989), pp. 479–483.

[253] J. A. Montgomery Jr, M. J. Frisch, J. W. Ochterski, and G. A. Petersson. “A complete basis set model chemistry. VI. Use of density functional geometries and frequencies”. In: *The Journal of chemical physics* 110.6 (1999), pp. 2822–2827.

[254] J. Rezác, K. E. Riley, and P. Hobza. “S66: A well-balanced database of benchmark interaction energies relevant to biomolecular structures”. In: *Journal of chemical theory and computation* 7.8 (2011), pp. 2427–2438.

[255] L. Goerigk, H. Kruse, and S. Grimme. “Benchmarking density functional methods against the S66 and S66x8 datasets for non-covalent interactions”. In: *ChemPhysChem* 12.17 (2011), pp. 3421–3433.

[256] A. Ambrosetti, A. M. Reilly, R. A. DiStasio, and A. Tkatchenko. “Long-range correlation energy calculated from coupled atomic response functions”. In: *The Journal of chemical physics* 140.18 (2014).

[257] P. P. Poier, T. Jaffrelot Inizan, O. Adjoua, L. Lagardere, and J.-P. Piquemal. "Accurate Deep Learning-Aided Density-Free Strategy for Many-Body Dispersion-Corrected Density Functional Theory". In: *The Journal of Physical Chemistry Letters* 13.19 (2022), pp. 4381–4388.

[258] B. D. Sellers, N. C. James, and A. Gobbi. "A comparison of quantum and molecular mechanical methods to estimate strain energy in druglike fragments". In: *Journal of chemical information and modeling* 57.6 (2017), pp. 1265–1275.

[259] M. J. Frisch, M. Head-Gordon, and J. A. Pople. "A direct MP2 gradient method". In: *Chemical Physics Letters* 166.3 (1990), pp. 275–280.

[260] M. Head-Gordon and T. Head-Gordon. "Analytic MP2 frequencies without fifth-order storage. Theory and application to bifurcated hydrogen bonds in the water hexamer". In: *Chemical Physics Letters* 220.1-2 (1994), pp. 122–128.

[261] W. J. Hehre, R. Ditchfield, and J. A. Pople. "Self—consistent molecular orbital methods. XII. Further extensions of Gaussian—type basis sets for use in molecular orbital studies of organic molecules". In: *The Journal of Chemical Physics* 56.5 (1972), pp. 2257–2261.

[262] A. McLean and G. Chandler. "Contracted Gaussian basis sets for molecular calculations. I. Second row atoms, Z= 11–18". In: *The Journal of chemical physics* 72.10 (1980), pp. 5639–5648.

[263] Y.-L. Liao and T. Smidt. "Equiformer: Equivariant graph attention transformer for 3d atomistic graphs". In: *arXiv preprint arXiv:2206.11990* (2022).

List of Figures

List of Tables

Force Field Parameterization in Molecular Simulation by Machine Learning Methods

Abstract

Molecular dynamics simulation allows for predictions of material properties, aiding in understanding, researching, and development of new materials. However, the accuracy of molecular force fields has been a long-standing limitation. Traditional force fields assign parameters based on discrete atom types, encoding all information about the atom's chemical environment. Higher accuracy requires more atom types, resulting in a proliferation of redundant parameters and low transferability. In this thesis, we introduce the Graph-Based Force Fields (GB-FFs) model, which employs graph neural networks to process directed molecular graphs and extract continuous atomic representations. These representations are then used to derive a set of force field parameters. GB-FFs model directly learn from quantum chemical energies and forces. In Chapter 1, we initially employ machine learning techniques to predict parameters for polynomial interpolation, demonstrating the effectiveness of neural networks in handling simple parameterization tasks. In Chapter 2, we represent molecules as strings and apply Natural Language Processing models to extract molecular fingerprints. In Chapter 3, we introduce Directed Graph Attention neTworks (D-GATs) to process directed molecular graphs, extracting molecular fingerprints and predicting molecular properties. In the final chapter, building upon the models from previous chapters, we propose the GB-FFs model, which achieves end-to-end molecular force field parameterization. Our methods have proven to be a reliable approach for optimizing and accelerating molecular force field parameterization. Currently, GB-FFs model has been tested and validated on the General AMBER Force Field (GAFF). Furthermore, our model's versatility allows for easy extension and application to other force fields in future research.

Keywords: graph neural networks, molecular force fields, molecular representation learning

---

Résumé

La simulation de dynamique moléculaire permet de prédire les propriétés des matériaux, contribuant ainsi à la compréhension, la recherche et le développement de nouveaux matériaux. Cependant, la précision des champs de force moléculaire a été une limitation depuis longtemps. Les champs de force traditionnels attribuent des paramètres selon des combinaisons de types d'atomes discrets, ce qui encode toutes les informations sur l'environnement chimique de l'atome. Une plus grande précision nécessite davantage de types d'atomes, ce qui entraîne une prolifération de paramètres redondants et une faible transférabilité. Dans cette thèse, nous présentons le modèle de Graph-Based Force Fields (GB-FFs), qui utilise des réseaux neuronaux de graphes pour traiter des graphes moléculaires dirigés et extraire des représentations atomiques. Ces représentations sont ensuite utilisées pour prédire tous les paramètres des champs de force. Le modèle GB-FFs apprend directement à partir des énergies et des forces. Dans le chapitre 1, nous utilisons initialement l'apprentissage automatique pour prédire les paramètres d'interpolation polynomiale, démontrant que les réseaux neuronaux peuvent réussir des tâches de paramétrisation. Dans le chapitre 2, nous représentons les molécules sous forme de chaînes et utilisons des modèles de traitement du langage naturel pour extraire des empreintes moléculaires. Dans le chapitre 3, nous introduisons Directed Graph Attention neTworks (D-GATs) pour traiter des graphes moléculaires dirigés, extraire des empreintes moléculaires et prédire des propriétés moléculaires. Dans le dernier chapitre, nous proposons le modèle GB-FFs, qui permet la paramétrisation des champs de force de bout en bout. Nos méthodes ont été démontrées comme une approche fiable pour optimiser et accélérer la paramétrisation des champs de force. Actuellement, le modèle GB-FFs a été testé et validé sur le General AMBER Force Field (GAFF). De plus, il pourra facilement être étendu et appliqué à d'autres champs de force également.

Mots clés : réseaux neuronaux graphiques, champs de forces moléculaires, apprentissage de représentation moléculaire

---

Laboratoire Jacques-Louis Lions
Sorbonne Université – Campus Pierre et Marie Curie – 4 place Jussieu – 75005 Paris – France