Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research



Yahia Fares University of Medea
Faculty of Sciences
Department of Mathematics and Computer Science

Final Master Project

Submitted for Obtaining The Master's Degree in Computer Science

Speciality: Applied Artificial Intelligence

Protein-Ligand Binding Affinity Prediction Using Graph Neural Networks

---

Prepared and Presented by:
ROUIS DJIHAD
CHIKHI ABDERRAOUF

Supervisor and Co-Supervisor:
Dr. MAZARI REDHA
Dr. HANTABLI HAMZA

2025 - 2026

Abstract

Protein-ligand binding affinity prediction is a central challenge in computational drug discovery, as it enables the rapid *in silico* screening of medical candidates before costly experimental validation. While classical physics-based methods and empirical scoring functions have long been applied to this problem, they suffer from either prohibitive computational cost or limited generalization across diverse molecular systems. Graph Neural Networks (GNNs) have emerged as a powerful alternative by representing molecules in their natural graph form and learning atomic interaction patterns directly from experimental data. However, the field of GNN architectures is vast, and it remains unclear which designs are most effective for structure-based binding affinity prediction.

This master's dissertation presents a systematic comparative evaluation of multiple GNN architectures on the protein-ligand binding affinity prediction task using the PDBbind dataset. We design a unified heterogeneous molecular graph representation that encodes the ligand and the protein pocket jointly, incorporating both covalent and non-covalent interaction information. All models are evaluated within an identical base architecture to ensure a fair and rigorous comparison, with the goal of identifying the most effective GNN design choices for structure-based drug discovery.

Keywords: Protein-ligand binding affinity, Graph Neural Networks, Drug discovery, PDBbind, Molecular graphs, Deep learning.

Résumé

La prédiction de l'affinité de liaison protéine-ligand est un défi central en découverte computationnelle de médicaments, car elle permet le criblage *in silico* rapide de candidats thérapeutiques avant une validation expérimentale coûteuse. Si les méthodes physiques classiques et les fonctions de score empiriques ont longtemps été appliquées à ce problème, elles souffrent soit d'un coût de calcul prohibitif, soit d'une capacité de généralisation limitée à des systèmes moléculaires diversifiés. Les réseaux de neurones sur graphes (GNNs) ont émergé comme une alternative puissante en représentant les molécules sous leur forme naturelle de graphe et en apprenant les patterns d'interaction atomique directement à partir de données expérimentales. Cependant, le paysage des architectures GNN est vaste, et il n'est pas encore clair quelles conceptions sont les plus efficaces pour la prédiction d'affinité de liaison basée sur la structure.

Ce mémoire présente une évaluation comparative systématique de plusieurs architectures GNN sur la tâche de prédiction de l'affinité de liaison protéine-ligand en utilisant la base de données PDBbind. Nous concevons une représentation hétérogène unifiée du graphe moléculaire qui encode conjointement le ligand et la poche de liaison protéique, en incorporant à la fois des informations d'interactions covalentes et non covalentes. Tous les modèles sont évalués dans une architecture de base identique afin de garantir une comparaison équitable et rigoureuse, avec pour objectif d'identifier les choix de conception GNN les plus efficaces pour la découverte de médicaments basée sur la structure.

Mots-clés : Affinité de liaison protéine-ligand, Réseaux de neurones sur graphes, Découverte de médicaments, PDBbind, Graphes moléculaires, Apprentissage profond.

ملخص

يعدّ التنبؤ بقوة الارتباط بين البروتين و الجزيئات تحديًا محوريًا في الإكتشاف الحسابي للأدوية، إذ تُتيح هذه العملية الفرز السريع للمرشحين العلاجيين افتراضيًا قبل إجراء التحقق التجريبي المُكلف. وعلى الرغم من أن الأساليب الفيزيائية الكلاسيكية ودوال التقييم التجريبية قد طُبِّقت على هذه المسألة منذ زمن بعيد، إلا أنها تعاني إما من تكلفة حسابية باهظة، أو من تعميم محدود عبر الأنظمة الجزيئية المتنوعة. وقد برزت شبكات الأعصاب الرسومية (sNNG) بصفتها بديلًا قويًا، من خلال تمثيل الجزيئات في صورتها الطبيعية كرسوم بيانية، وتعلّم أنماط التفاعل الذري مباشرةً من البيانات التجريبية. غير أن مجال الشبكات الرسومية واسع للغاية، ولا يزال غير واضح أي نموذج هو الأكثر فعالية في التنبؤ بقوة الارتباط استنادًا إلى البنية الثلاثية الأبعاد.

تقدم هذه المذكرة مقارنة منهجية لعدة بنيات من شبكات الأعصاب الرسومية على مهمة التنبؤ بقوة الارتباط بين البروتين والجزيء باستخدام قاعدة بيانات dnibBDP. حيث نُصمم تمثيلًا موحدًا وغير متجانس للمركب الجزيئي يُشفِّر بصورة مشتركة الجزيئات وموضع الارتباط البروتيني، مع دمج معلومات التفاعلات التساهمية وغير التساهمية. يتم تقييم جميع النماذج ضمن بنية أساسية موحدة لضمان مقارنة عادلة ودقيقة، بهدف تحديد خيارات التصميم الأكثر فعالية في شبكات NNG لاكتشاف الأدوية المبني على البنية الثلاثية الأبعاد.

الكلمات المفتاحية: تقارب الارتباط بين البروتين و الجزيء، شبكات الأعصاب الرسومية، اكتشاف الأدوية، dnibBDP، الرسوم البيانية الجزيئية، التعلم العميق.

Contents

Abstract

Résumé

ملخص

List of Abbreviations

General Introduction **1**

1 Biological Background and Context **4**

1.1 Introduction 4

1.2 Proteins 4

1.2.1 Definition and Structure 4

1.2.2 Role in the Human Body 6

1.3 Ligands 7

1.3.1 Definition 7

1.3.2 Types of Ligands 8

1.4 Protein–Ligand Binding 9

1.4.1 Definition of Binding Affinity 9

1.4.2 The Concept of ΔG (Free Energy) 9

1.4.3 How Binding Works: Active Site and Interactions 10

1.5 Importance in Drug Discovery 11

1.5.1 The Drug Discovery Pipeline 11

1.5.2 Role of Computational Methods 12

1.5.3 Why Prediction Matters 12

1.6 Datasets Used in Binding Affinity Research 13

1.6.1 PDBbind Dataset 13

1.6.2 Data Challenges 13

1.7 Case Studies 14

1.8 Conclusion 15

2 Deep Learning and Graph Neural Networks **17**
2.1 Introduction . . . 17
2.2 Machine Learning Fundamentals . . . 18
2.2.1 Supervised Learning . . . 18
2.2.2 Evaluation Metrics . . . 18
2.3 Deep Learning and Neural Networks . . . 19
2.3.1 Artificial Neural Networks . . . 20
2.3.2 Activation Functions . . . 21
2.4 Graph Theory Fundamentals . . . 22
2.4.1 Definition of a Graph . . . 22
2.4.2 Graph Representations and the Need for Graph Neural Networks . . . 23
2.5 Graph Neural Networks (GNNs) . . . 23
2.5.1 Message Passing Framework . . . 24
2.5.2 Graph Convolutional Networks (GCN) . . . 25
2.5.3 Graph Attention Networks (GAT) . . . 26
2.5.4 Graph Isomorphism Networks (GIN/GINE) . . . 27
2.5.5 GraphSAGE . . . 28
2.5.6 TransformerConv . . . 28
2.5.7 Equivariant Graph Neural Networks (EGNN) . . . 29
2.5.8 GENeralized Aggregation Networks (GENConv) . . . 30
2.5.9 Principal Neighbourhood Aggregation (PNA) . . . 31
2.5.10 General, Powerful, Scalable Graph Transformer (GPS) . . . 32
2.5.11 Chebyshev Graph Convolution (ChebConv) . . . 33
2.5.12 Simple Graph Convolution (SGC) . . . 34
2.6 Molecular Graphs . . . 35
2.6.1 Representing Molecules as Graphs . . . 35
2.6.2 Node Features . . . 35
2.6.3 Edge Features . . . 36
2.6.4 RBF Encoding of Interatomic Distances . . . 37
2.7 Related Works . . . 38
2.8 Conclusion . . . 40

3 Methodology **43**
3.1 Introduction . . . 43
3.2 Project Objectives . . . 44
3.3 System Architecture Overview . . . 44
3.4 Dataset . . . 45
3.4.1 PDBbind Dataset . . . 45
3.4.2 Data Filtering . . . 46

3.4.3 Label Distribution . . . 46
3.5 Graph Construction . . . 47
3.5.1 Ligand Graph . . . 47
3.5.2 Pocket Graph . . . 49
3.5.3 Interaction Edges . . . 50
3.5.4 Edge Feature Unification . . . 51
3.6 Dataset Class and Label Processing . . . 52
3.6.1 Label Normalization . . . 52
3.6.2 Train/Validation/Test Split . . . 53
3.7 Model Architecture . . . 53
3.7.1 Input Projection . . . 54
3.7.2 GNN Layers . . . 55
3.7.3 Global Pooling . . . 55
3.7.4 MLP Head . . . 56
3.8 Training Pipeline . . . 56
3.8.1 Loss Function . . . 56
3.8.2 Optimizer and Scheduler . . . 57
3.8.3 Checkpointing . . . 57
3.9 Conclusion . . . 58

4 Experimental Results and Discussion **60**
4.1 Introduction . . . 60
4.2 Development Environment . . . 60
4.2.1 Technologies and Tools . . . 60
4.2.2 Python . . . 60
4.2.3 PyTorch . . . 61
4.2.4 PyTorch Geometric . . . 61
4.2.5 RDKit . . . 61
4.2.6 Google Colab . . . 61
4.2.7 Kaggle . . . 62
4.3 Experimental Setup . . . 62
4.3.1 Training Parameters . . . 62
4.3.2 Evaluation Protocol . . . 63
4.4 Performance Comparison Table . . . 64
4.5 Model Results . . . 65
4.6 Best Model Analysis . . . 67
4.6.1 Training Curves . . . 67
4.6.2 Predicted vs Actual Scatter Plot . . . 68
4.6.3 Error Distribution . . . 68

4.6.4 Comparative Prediction Quality Across Top Models 69
4.7 Comparative Analysis with Related Works 70
4.8 Analysis of Factors Influencing Results 71
4.9 Challenges and Obstacles 73
4.10 Conclusion 74

General CONCLUSION **76**

Bibiography **78**

Webography **81**

List of Figures

1.1 chemical structure of an amino acid[1] 5
1.2 Structure of a protein 6
1.3 A 3D graph representation of a ligand molecule where atoms represented as nodes and bonds represented as edges[17]. 8
1.4 3D visualization of the ligand (spheres) occupying the complementary binding pocket (surface rendering) of the target protein[34]. 10
1.5 The sequential stages of the protein-ligand drug discovery pipeline[3]. . . . 11

2.1 Artificial neural network diagram[10]. 21
2.2 The end-to-end Graph Neural Network (GNN) pipeline for graph-level prediction [14]. 24
2.3 Graph Neural Network message passing aggregation update diagram[12]. . 25
2.4 Sixteen Gaussian basis functions spanning the distance range 0–6 Å[18]. . . 38

3.1 Overview of the proposed pipeline for protein–ligand binding affinity prediction. 45
3.2 Histogram of binding free energy (ΔG) values in the filtered PDBbind dataset. The distribution is approximately Gaussian and centered near −8.73 kcal/mol. 47
3.3 Three-dimensional ball-and-stick representation of a ligand molecule from the PDBbind dataset[1]. 49
3.4 Three-dimensional ball-and-stick representation of the protein binding pocket extracted from the PDB file. Color coding: grey (Carbon), red (Oxygen), blue (Nitrogen), yellow (Sulfur)[1]. 50
3.5 Visualization of the unified heterogeneous molecular graph for a protein-ligand complex from PDBbind[15]. 51
3.6 Base GNN architecture for binding affinity prediction. 54

4.1 Comparison of the 13 evaluated models based on their Test Pearson correlation coefficient (R). Models are sorted in descending order of performance, from highest to lowest Test Pearson R. 65

4.2 Radar chart comparing the six best-performing models across four evaluation metrics: RMSE, MAE, Pearson correlation coefficient (R), and Spearman rank correlation coefficient 66
4.3 GENConv training curves 67
4.4 Scatter plot of GENConv predicted vs experimental ΔG (kcal/mol) on the test set. Each point represents one protein-ligand complex. The dashed line indicates perfect prediction. 68
4.5 Distribution of prediction errors (predicted − experimental ΔG) for the GENConv model on the test set. 69
4.6 Distribution of prediction errors (predicted − experimental ΔG) for the GENConv model on the test set. 70
4.7 Comparison of the average Test Pearson correlation coefficient (R) for models with edge features and models without edge features. 72

List of Tables

2.1 Node feature encoding used for all atoms in the molecular graph. 36
2.2 Summary of related works on GNN-based binding affinity prediction. . . . 40

3.1 Node feature encoding for ligand and pocket atoms. 48
3.2 Composition of the unified 21-dimensional edge feature vector. 52
3.3 GNN operators evaluated in this work. 55

4.1 Training hyperparameters table 62
4.2 Results of models performance 64
4.3 Comparison of our best model with related works on PDBbind. 71

List of Abbreviations

AI	Artificial Intelligence
ADMET	Absorption, Distribution, Metabolism, Excretion, and Toxicity
ANN	Artificial Neural Network
BCR-ABL	Breakpoint Cluster Region - Abelson
CADD	Computer-Aided Drug Discovery
CASF	Comparative Assessment of Scoring Functions
ChebConv	Chebyshev Graph Convolution
CI	Concordance Index
CNN	Convolutional Neural Network
COVID-19	Coronavirus Disease 2019
EGNN	Equivariant Graph Neural Network
FEP	Free Energy Perturbation
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GELU	Gaussian Error Linear Unit
GENConv	Generalized Aggregation Network Convolution
GIN	Graph Isomorphism Network
GINE	Graph Isomorphism Network with Edge features
GNN	Graph Neural Network
GPCR	G Protein-Coupled Receptor
GPS	General, Powerful, Scalable Graph Transformer
GPU	Graphics Processing Unit

HTS	High-Throughput Screening
IC_{50}	Half-Maximal Inhibitory Concentration
KIBA	Kinase Inhibitor BioActivity
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
MM/GBSA	Molecular Mechanics / Generalized Born Surface Area
MM/PBSA	Molecular Mechanics / Poisson-Boltzmann Surface Area
MPNN	Message Passing Neural Network
Mpro	Main Protease
PDB	Protein Data Bank
PDBbind	Protein Data Bank Binding Affinity Database
PNA	Principal Neighbourhood Aggregation
PyG	PyTorch Geometric
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SAGE	Sample and Aggregate
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SD	Standard Deviation
SDF	Structure Data File
SGC	Simple Graph Convolution
SGD	Stochastic Gradient Descent
SMILES	Simplified Molecular Input Line Entry System
TPU	Tensor Processing Unit
WL	Weisfeiler-Lehman

General Introduction

Context and Motivation

The discovery of new medical compounds is one of the most complex and resource-intensive studies in modern science. The traditional drug discovery pipeline requires an average of ten to fifteen years and over one billion dollars to bring a single drug from initial discovery to clinical approval, with the vast majority of candidate compounds failing at some stage of development due to insufficient efficacy, toxicity, or poor pharmacokinetic properties [3]. A critical bottleneck in this process is the early-stage identification of compounds that bind strongly and selectively to a target protein, this challenge known as binding affinity prediction. The binding affinity between a small molecule ligand and its target protein, quantified as the Gibbs free energy of binding ΔG, determines whether a compound is likely to produce a measurable biological effect at clinically relevant concentrations. Accurate computational prediction of this quantity would allow researchers to rapidly screen millions of candidate compounds *in silico*, dramatically reducing the time and cost of experimental screening campaigns.

Classical approaches to binding affinity prediction include physics-based methods such as molecular dynamics simulations and free energy perturbation, which provide high accuracy but are computationally expensive for large-scale screening. Empirical scoring functions, which approximate binding affinity as a weighted sum of physically motivated interaction terms, offer faster predictions but sacrifice accuracy and generalization across diverse protein families. The emergence of machine learning has introduced a new generation of data-driven methods that learn the relationship between molecular structure and binding affinity directly from experimental data, without requiring explicit specification of the physical interaction terms [17].

Problem Statement

Despite significant progress in machine learning-based binding affinity prediction, several fundamental challenges remain unresolved. Sequence-based methods that represent proteins

as amino acid sequences and ligands as SMILES strings discard the three-dimensional structural information that is physically responsible for binding. Grid-based methods that represent molecular structures as three-dimensional voxel grids are computationally expensive and sensitive to the orientation of the input structure. Neither of these representations captures the essentially graph-structured nature of molecules, where atoms are nodes and chemical bonds are edges, in a way that is simultaneously expressive, computationally efficient, and invariant to irrelevant transformations such as molecular rotation and translation.

Objectives

This work addresses these challenges by developing a unified framework for comparative evaluation of GNN architectures on protein-ligand binding affinity prediction. The specific objectives of this thesis are four-directional. First, we construct a heterogeneous molecular graph representation that jointly encodes the ligand and the protein binding pocket as a single graph, capturing covalent bonds within each component and non-covalent spatial interactions at the binding interface through RBF-encoded distance features. Second, we implement a shared base architecture comprising a node embedding layer, four GNN layers with residual connections, dual pooling, and an MLP regression head, within which GNN convolution operators are evaluated under identical conditions. Third, we train and evaluate all models on the PDBbind dataset using a custom RangeAwareLoss function that prevents prediction collapse, and report test performance using four complementary metrics: RMSE, MAE, Pearson R, and Spearman R. Fourth, we analyze the results to identify which architectural properties most consistently drive performance improvements, providing actionable insights for future GNN-based drug discovery systems.

Thesis Structure

The remainder of this thesis is organized as follows. Chapter 1 presents the biological and chemical background of protein-ligand binding, including the thermodynamic basis of binding affinity and the PDBbind dataset. Chapter 2 provides the theoretical foundations of machine learning, deep learning, graph theory, and the thirteen GNN architectures evaluated in this work, together with a review of related works in the literature. Chapter 3 describes the complete methodology, including the molecular graph construction pipeline, the shared base architecture, and the training configuration. Chapter 4 presents the experimental results, comparative analysis, and discussion of factors influencing model performance, followed by directions for future work. The thesis concludes with a general conclusion summarizing the main findings and their implications for computational drug discovery.

Biological Background and Context

Chapter 1

Biological Background and Context

1.1 Introduction

The discovery of new drugs is one of the most challenging and resource-intensive processes in modern science. At the heart of this process lies a fundamental question: how strongly does a candidate molecule bind to its target protein? Answering this question accurately and efficiently has become a central goal of computational biology and artificial intelligence research.

This chapter introduces the biological foundations necessary to understand the problem addressed in this thesis. We begin by examining the structure and function of proteins, then describe the nature of ligands and the molecular interactions that control binding. We then explain the concept of binding affinity and its quantification through the Gibbs free energy ΔG, before presenting the role of computational methods in drug discovery. Finally, we introduce the PDBbind dataset used in our work and outline the challenges it presents.

1.2 Proteins

1.2.1 Definition and Structure

Proteins are biological macromolecules that serve as the fundamental building blocks of life, critical to almost every biological process. The term "protein" derives from the Greek word *proteios*, meaning "of first importance," reflecting their essential role in all living organisms. Proteins are composed of one or more polypeptides which are a linear chains of amino acids linked together by peptide bonds[1]. Although cells may contain dozens of amino acid types, but only 20 standard amino acids are commonly found in proteins, each distinguished by a unique side chain known as the R group, which determines its chemical properties and functional role[1].

Proteins are polymers composed of amino acid monomers linked together by peptide bonds. Each amino acid consists of a central carbon atom (the α-carbon) bonded to four groups: an amino group ($-NH_2$), a carboxyl group ($-COOH$), a hydrogen atom, and a variable side chain known as the R-group that determines the chemical identity and properties of each amino acid [1]. There are 20 standard amino acids found in naturally occurring proteins, differing only in the structure of their R-group. These R-groups vary widely in their chemical properties, some are nonpolar and hydrophobic (such as alanine, valine, and leucine), some are polar and uncharged (such as serine, threonine, and asparagine), some carry a positive charge at physiological pH (such as lysine, arginine, and histidine), and some carry a negative charge (such as aspartate and glutamate) [1].

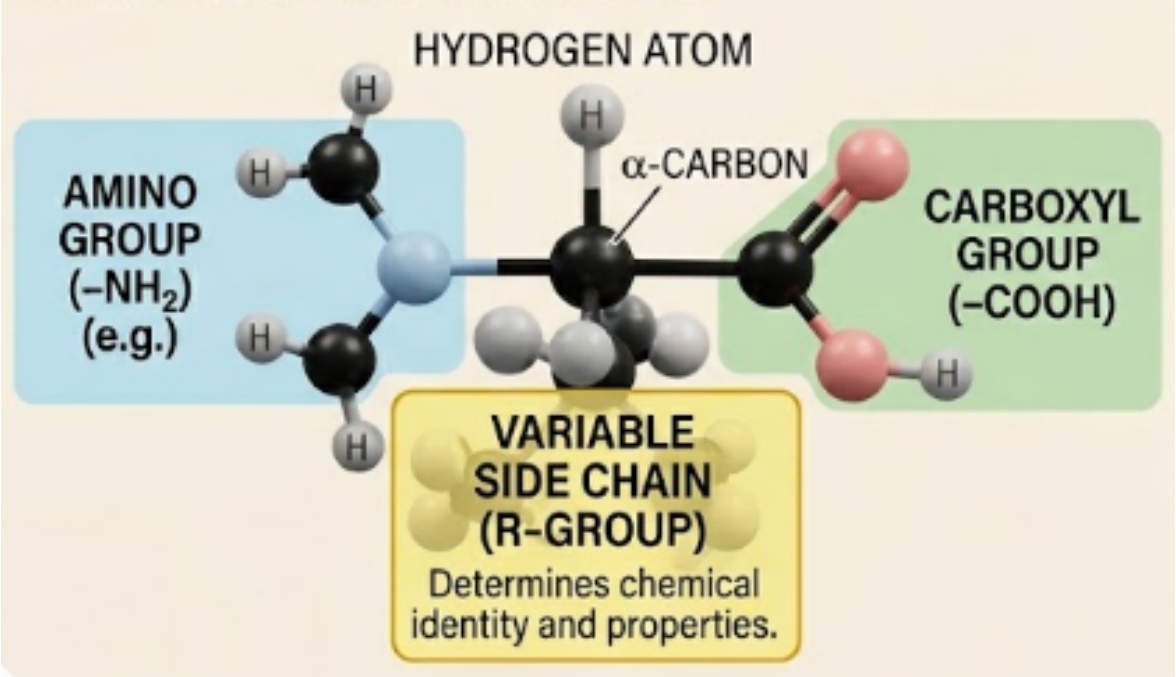


Figure 1.1: chemical structure of an amino acid[1]

Protein structure is organized into four hierarchical levels[2]. The **primary structure** is the linear sequence of amino acids in the polypeptide chain, this sequence is the ultimate determinant of the protein's three-dimensional shape and function. The **secondary structure** refers to localized folding patterns within the peptide backbone, primarily alpha-helices and beta-pleated sheets, stabilized by hydrogen bonds between the amide N–H and carbonyl C=O groups[2]. The **tertiary structure** is the complete three-dimensional structure of all atoms in a single polypeptide, including side chains, defining the overall shape and the precise geometry of the active site where ligand binding occurs[2]. When a protein comprises multiple polypeptide chains (subunits), their spatial arrangement constitutes the **quaternary structure**, stabilized by non-covalent forces including hydrogen bonds, electrostatic attractions, and hydrophobic interactions[2].

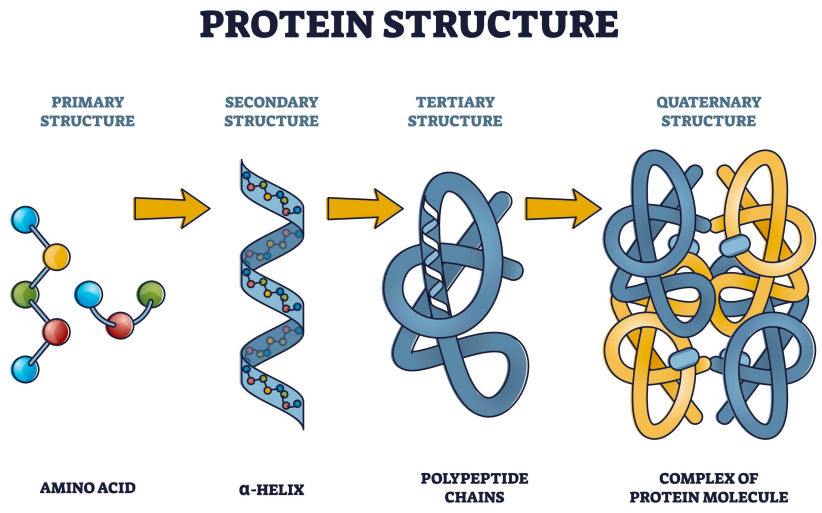


Figure 1.2: Structure of a protein showing polypeptide chain organization[2].

The chemical diversity of amino acid side chains plays a critical role in protein behavior. Nonpolar side chains tend to be buried in the hydrophobic core of the folded protein, while polar and charged residues are typically exposed to the aqueous environment[1]. Certain amino acids participate in key stabilizing interactions: ionic bonds form between oppositely charged side chains, and disulfide bridges form covalent bonds between cysteine residues[1]. These interactions are essential for maintaining the protein's three-dimensional structure and, consequently, its biological function. The critical importance of primary structure is illustrated by sickle-cell anemia, a genetic disorder caused by a single amino acid substitution, glutamic acid replaced by valine at position six of the beta-globin chain, which is sufficient to alter hemoglobin's shape and impair its ability to transport oxygen.

In computational drug discovery, proteins are represented at multiple levels of detail. At the simplest level, a protein is encoded as a one-dimensional sequence of amino acid residues. At higher resolution, the three-dimensional atomic coordinates obtained from X-ray crystallography or cryo-electron microscopy, deposited in the Protein Data Bank (PDB), provide a complete spatial description[1]. Intermediate representations such as contact maps, which encode pairwise distances between alpha-carbon atoms of residues, have been widely used in deep learning models for binding affinity prediction as they capture structural information derivable from sequence alone[17].

1.2.2 Role in the Human Body

Proteins perform a vast and diverse array of essential functions in the human body. As enzymes, they catalyze virtually all biochemical reactions that sustain life, from metabolic processes to DNA replication and repair[1]. As structural proteins, they provide mechanical

support to cells and tissues, collagen reinforces connective tissue, and keratin forms hair and nails[1]. Transport proteins such as hemoglobin carry oxygen through the bloodstream, while membrane proteins regulate the passage of ions and molecules across cell membranes [1]. Signaling proteins, including hormones and receptors, transmit information between cells and coordinate physiological responses [1].

Many diseases arise from abnormalities in the amino acid sequence or folding of specific proteins. Huntington's disease results from an expanded glutamine repeat in the huntingtin protein, causing progressive neurodegeneration. Cystic fibrosis arises from mutations in the CFTR chloride channel protein, impairing ion transport and producing thick mucus in the lungs and other organs. Mutations in the tumor suppressor protein p53 one of the most commonly mutated genes in human cancers disrupt cell cycle regulation and DNA repair, promoting uncontrolled cell proliferation. These examples underscore the central role of proteins in both normal physiology and disease, and explain why proteins constitute the primary targets of therapeutic intervention in modern drug discovery.

G protein-coupled receptors (GPCRs), which represent the largest family of membrane receptor proteins, are particularly important drug targets. They mediate cellular responses to hormones, neurotransmitters, and sensory stimuli, and account for approximately one-third of all approved therapeutic drugs[3]. Kinases, another major class of drug target proteins, regulate cell signaling pathways and are frequently mutated or overexpressed in cancer[3]. The diversity of protein function and the specificity of protein–ligand interactions make accurate structural and functional characterization of proteins a prerequisite for rational drug design.

1.3 Ligands

1.3.1 Definition

In the context of biochemistry and drug discovery, a ligand is any molecule that binds to a specific site on a target protein[7]. Ligands are typically small organic molecules, also referred to as small molecules or drug-like molecules, that interact with proteins through non-covalent forces[7]. The binding of a ligand to its target protein can modulate the protein's activity, either activating or stopping its function[3]. In computational drug discovery, ligands are commonly represented as SMILES strings which is a linear text notation that encodes the molecular graph of a compound or as two-dimensional molecular graphs in which atoms serve as nodes and chemical bonds serve as edges[17].

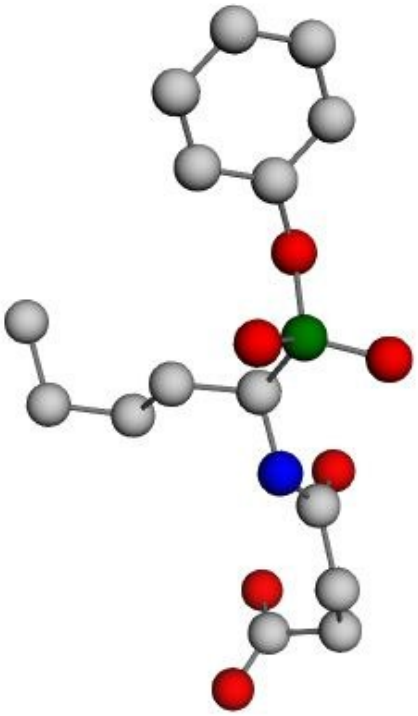


Figure 1.3: A 3D graph representation of a ligand molecule where atoms represented as nodes and bonds represented as edges[17].

1.3.2 Types of Ligands

Ligands can be classified according to their effect on the target protein[3]. **Agonists** are ligands that bind to a receptor and activate it, mimicking the effect of the endogenous molecule[3]. **Antagonists** bind to the same site but block activation, inhibiting the receptor's response[3]. **Inhibitors** more broadly refer to ligands that reduce the activity of an enzyme or receptor, and include competitive inhibitors that bind at the active site and allosteric inhibitors that bind at a distinct regulatory site to indirectly reduce activity[3]. Cofactors and enzymatic substrates represent additional ligand types found in structural databases.

In the context of binding affinity datasets, ligands are often further categorized by their measured potency metrics: dissociation constant (K_d), inhibition constant (K_i), and half-maximal inhibitory concentration (IC_{50}) are the most commonly reported values, each reflecting a slightly different aspect of the ligand–protein interaction[1].

1.4 Protein–Ligand Binding

1.4.1 Definition of Binding Affinity

Binding affinity refers to the strength of the non-covalent interaction between a protein and a ligand[7]. It is one of the most important parameters in drug discovery, as it determines how effectively a drug candidate will occupy its target at physiologically relevant concentrations[3]. Binding affinity is most commonly expressed as the dissociation constant K_d, which represents the concentration of ligand at which half of the protein binding sites are occupied at equilibrium[1]. A lower K_d value indicates a stronger, higher-affinity interaction.

Related measures include the inhibition constant K_i, which quantifies the potency of an inhibitor, and the half-maximal inhibitory concentration IC_{50}, which reflects the concentration of a compound required to inhibit a biological process by 50%[1]. These values are often converted to their negative logarithmic forms: $\text{p}K_d$, $\text{p}K_i$, and pIC_{50}, so that higher values correspond to stronger binding, facilitating model training and comparison[17].

1.4.2 The Concept of ΔG (Free Energy)

The thermodynamic basis of protein–ligand binding is described by the change in Gibbs free energy ΔG upon complex formation[2]. A negative ΔG indicates that binding is thermodynamically favorable and will occur spontaneously[2]. The relationship between binding affinity and free energy is given by:

$$\Delta G = RT \ln K_d \tag{1.1}$$

where R is the universal gas constant and T is the absolute temperature[2]. This equation shows that stronger binding (lower K_d) corresponds to a more negative ΔG.

In relative binding free energy (RBFE) calculations, the difference in binding free energy between two ligands $\Delta\Delta G$ is computed, and can be converted to a $\text{p}K_i$ value using:

$$\text{p}K_i = -\frac{\Delta\Delta G + \Delta G_{\text{ref}}}{RT \ln 10} \tag{1.2}$$

where ΔG_{ref} is the absolute binding free energy of a reference compound[2]. These free energy-based methods are among the most accurate computational approaches for binding affinity prediction, though their high computational cost limits their application to small compound sets[3].

1.4.3 How Binding Works: Active Site and Interactions

Protein–ligand binding occurs at a specific region of the protein surface known as the **active site** or **binding pocket**, which is a cavity or cleft whose shape and chemical properties are complementary to those of the ligand[7]. The binding interaction is driven by a combination of non-covalent forces[2]. **Van der Waals interactions** arise from transient dipole-induced dipole attractions between atoms in close proximity[2]. **Electrostatic interactions** occur between charged or polar groups on the protein and ligand[2]. **Hydrogen bonds** form between hydrogen bond donors and acceptors on both molecules and contribute significantly to binding specificity[7]. **Hydrophobic interactions** drive the burial of non-polar groups away from the aqueous solvent into the hydrophobic core of the binding pocket[2]. In some cases, π**-stacking interactions** between aromatic rings also contribute to binding stability[7].

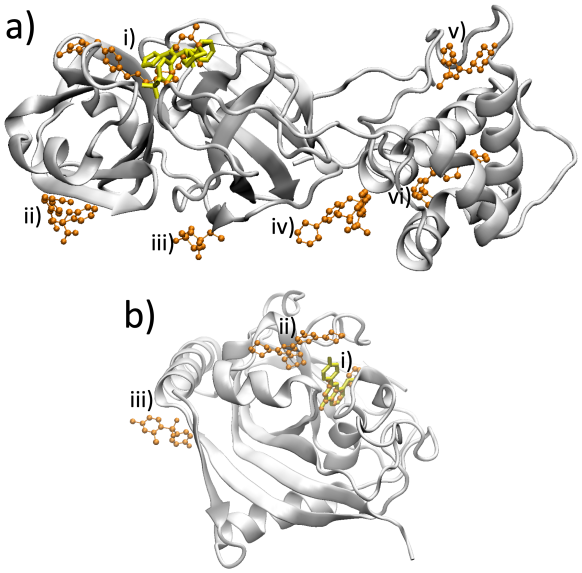


Figure 1.4: 3D visualization of the ligand (spheres) occupying the complementary binding pocket (surface rendering) of the target protein[34].

The total binding free energy can be approximated as the sum of these contributions:

$$\Delta G_{\text{binding}} = \Delta E_{\text{VdW}} + \Delta E_{\text{el}} + \Delta E_{\text{H-bond}} + \Delta G_{\text{sol}} \tag{1.3}$$

where each term represents the contribution of van der Waals, electrostatic, hydrogen bonding, and solvation effects respectively[2]. Understanding these interactions at the atomic level is essential for the rational design of potent and selective drug candidates, and motivates the use of three-dimensional structural representations in computational binding affinity models[17].

1.5 Importance in Drug Discovery

1.5.1 The Drug Discovery Pipeline

The discovery and development of a new small-molecule drug is a lengthy and expensive process, typically requiring over a decade and billions of dollars before a drug reaches the market[3]. The pipeline begins with **target identification**, where a protein implicated in a disease is selected as a therapeutic target. This is followed by **hit identification**, commonly performed through high-throughput screening (HTS), in which large libraries of compounds are tested for activity against the target[4]. Active compounds identified in this stage, known as hits, are then refined through **hit-to-lead optimization** to improve their potency, selectivity, and drug-like properties[3]. The lead compound is subsequently optimized further with respect to absorption, distribution, metabolism, excretion, and toxicity (ADMET) properties before advancing to preclinical and clinical trials[3].

Computer-aided drug discovery (CADD) plays an important role at multiple stages of this pipeline. Structure-based virtual screening can rapidly identify potential **hits** from large compound libraries[4]. Binding affinity prediction models support lead optimization by guiding the selection of compounds with improved potency[7]. Computational prediction of ADMET properties reduces the risk of late-stage attrition by flagging problematic compounds early in the process[3].

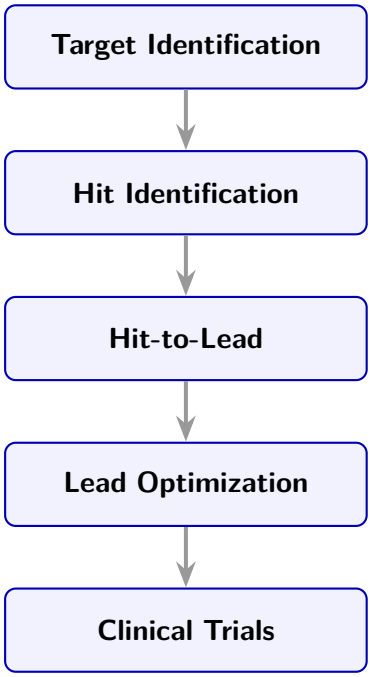


Figure 1.5: The sequential stages of the protein-ligand drug discovery pipeline[3].

1.5.2 Role of Computational Methods

Computational methods for predicting protein–ligand binding affinity span a wide spectrum of accuracy and computational cost[17]. Physics-based methods such as free energy perturbation (FEP) and molecular mechanics approaches (MM/GBSA, MM/PBSA) offer high accuracy by explicitly modeling the thermodynamics of binding, but are computationally expensive and impractical for large-scale screening[2]. Molecular docking methods provide a faster alternative by predicting the binding pose of a ligand within the protein active site and estimating the binding affinity using empirical, knowledge-based, or force-field-based scoring functions, though their accuracy is often limited[8].

Machine learning and deep learning methods have emerged as powerful alternatives that combine computational efficiency with competitive predictive accuracy[17]. These data-driven approaches learn to predict binding affinities directly from molecular representations including SMILES strings, amino acid sequences, molecular graphs, contact maps, and three-dimensional atomic coordinates without requiring explicit physical modeling[10]. Deep learning architectures including convolutional neural networks (CNNs), recurrent neural networks (RNNs), graph neural networks (GNNs), and Transformer-based models have all been applied to this task, each offering distinct advantages in terms of feature extraction and representation learning[17].

1.5.3 Why Prediction Matters

The ability to accurately predict protein–ligand binding affinity *in silico* is of enormous practical value in drug discovery[3]. Experimental measurement of binding affinity, while reliable, is slow and resource-intensive, making it impractical to test the vast chemical spaces explored in modern drug discovery campaigns[4]. Computational prediction methods can screen millions of compounds orders of magnitude faster than experimental assays, enabling researchers to prioritize the most promising candidates for synthesis and testing[17].

The economic impact is substantial. For example, using relative binding free energy calculations to evaluate a chemical library of 5,000 compounds at an average cost of 8 GPU hours per calculation at $2.50 per GPU hour would cost approximately $100,000. By contrast, using an active learning framework guided by a predictive model to label only 300 strategically selected compounds reduces this cost to approximately $6,000, while achieving comparable predictive performance. This dramatic reduction in cost and time directly accelerates the drug discovery process and increases the probability of identifying effective therapeutic candidates[3].

1.6 Datasets Used in Binding Affinity Research

1.6.1 PDBbind Dataset

The PDBbind database is one of the most widely used datasets for training and evaluating machine learning and deep learning models for protein–ligand binding affinity prediction[1]. It is a selected subset of the Protein Data Bank (PDB) that collects experimentally determined three-dimensional structures of protein–ligand complexes together with their measured binding affinities, expressed as K_d, K_i, or IC_{50} values[1].

The database is organized into two subsets: the **general set**, which contains all collected protein–ligand complexes, and the **refined set**, which applies more stringent quality criteria[1]. To be included in the refined set, a complex must satisfy several conditions: the crystal structure resolution must be 2.5 Å or better, the ligand must be non-covalently bound without significant steric clashes, only equilibrium binding constants (K_d or K_i) are accepted (IC_{50} values are excluded due to their assay dependence), and complexes with missing backbone fragments near the binding site or extreme binding affinity values are removed[1].

The current release (PDBbind 2020) contains 19,443 protein–ligand complexes[1]. Earlier versions 2013, 2016, and 2018 containing 11,184, 13,283, and 16,126 complexes respectively, are also widely used in the literature[17].

A core subset known as the **CASF benchmark** is derived from the PDBbind refined set by clustering proteins by sequence similarity and selecting representative complexes spanning a range of binding affinities[1]. The CASF-2016 benchmark contains 285 protein–ligand complexes from 57 protein clusters and serves as the de facto standard for evaluating the scoring, ranking, and docking performance of computational binding affinity models[1].

1.6.2 Data Challenges

Despite the availability of large and well-curated datasets, several challenges complicate the development and evaluation of binding affinity prediction models[17]. One fundamental issue is the **heterogeneity of affinity measurements** across datasets. Different experimental assays report binding affinity in terms of K_d, K_i, or IC_{50}, which reflect subtly different aspects of the binding interaction and cannot be directly compared without careful normalization[1]. Combining measurements from multiple assay types as done in the KIBA dataset, which integrates K_i, K_d, and IC_{50} values into a unified KIBA score introduces uncertainty that is not fully accounted for and can limit the maximum achievable prediction accuracy[17].

Structural data quality presents additional challenges[17]. Some protein–ligand complexes in public databases contain covalently bound ligands, missing backbone or side chain fragments near the binding site, or ternary complexes involving multiple binding partners, all of which

complicate the extraction of meaningful binding affinity signals[1]. The PDBbind refined set explicitly excludes covalently bound ligands and complexes with resolution above 2.5 Å to address these issues, but such filtering reduces the available training data[1].

Dataset bias is another significant concern[17]. Published binding affinity data tends to be skewed toward potent binders, as weak or non-binding compounds are less likely to be reported in the literature[3]. This distributional bias can cause models to perform poorly on weak binders. Furthermore, when training sets and test sets share similar protein sequences or ligand scaffolds, model performance is artificially inflated due to data leakage[17]. Random splits of datasets can allow closely related compounds to appear in both training and test sets, overestimating the model's ability to generalize to genuinely novel targets or chemotypes. Scaffold-based or sequence-identity-based splits provide a more realistic assessment of generalization performance[17].

1.7 Case Studies

The importance of accurate binding affinity prediction is illustrated by several landmark applications in the pharmaceutical industry.

Kinase inhibitor design represents one of the most successful areas of structure-based drug discovery. Protein kinases are enzymes that regulate cell signaling pathways and are implicated in many cancers. The development of imatinib (Gleevec), a BCR-ABL kinase inhibitor, was guided by structural knowledge of the kinase binding pocket and represented a breakthrough in targeted cancer therapy. Machine learning models trained on binding affinity data for kinase–inhibitor pairs, such as those evaluated on the Davis and KIBA datasets, have demonstrated the ability to predict affinity patterns across hundreds of kinase targets simultaneously [17].

Protease inhibitors for HIV represent another class of drugs whose development was deeply informed by binding affinity data. HIV protease inhibitors were among the first drugs to be designed using computational structure-based methods, and their success contributed to transforming HIV from a fatal disease to a manageable chronic condition [3].

SARS-CoV-2 main protease (M^{pro}) inhibitors have been a major focus during the COVID-19 pandemic. The COVID Moonshot project collected binding affinity data for thousands of compounds against M^{pro} and made this data publicly available, fueling the development and benchmarking of new machine learning models for affinity prediction [17].

These examples illustrate that binding affinity prediction is not merely an academic exercise but directly supports the development of life-saving medicines and the optimization of existing therapeutic agents.

1.8 Conclusion

This chapter has provided the foundational background necessary to understand the problem of protein–ligand binding affinity prediction in the context of drug discovery. Proteins and ligands interact through specific non-covalent forces at defined binding sites, and the strength of this interaction (quantified as binding affinity) is a central metric in the evaluation of drug candidates. The thermodynamic framework of free energy provides a rigorous basis for understanding and computing binding affinities, while the drug discovery pipeline illustrates the practical context in which accurate affinity prediction is most valuable.

Computational methods have evolved from physics-based free energy calculations to modern deep learning architectures capable of processing diverse molecular representations at scale. The availability of curated datasets such as PDBbind, Davis, and KIBA has enabled the development and benchmarking of these methods, though challenges related to data quality, heterogeneity, and bias remain active areas of research. The following chapters build upon this foundation by presenting the specific deep learning methods, experimental frameworks, and evaluation strategies employed in this work.

Deep Learning and Graph Neural Networks

Chapter 2

Deep Learning and Graph Neural Networks

2.1 Introduction

The accurate prediction of protein–ligand binding affinity represents one of the most critical challenges in computational drug discovery. While traditional methods have made significant contributions to this field, they often fail to capture the complex atom-level interactions between proteins and ligands, limiting their predictive accuracy. The emergence of deep learning has introduced powerful new approaches that can automatically extract meaningful representations from raw molecular data, offering a faster and more scalable alternative to classical physics-based methods [17]. These methods can be broadly categorized into sequence-based models, which operate on one-dimensional representations such as amino acid sequences and SMILES strings, graph-based models, which represent molecules as graphs to capture their topological structure, and multimodal models, which combine multiple input representations to achieve richer feature extraction [17].

This chapter presents the theoretical foundations underlying the methods employed in this work. Section 2.2 introduces the core concepts of machine learning relevant to binding affinity prediction, including supervised learning and evaluation metrics. Section 2.3 covers the principles of deep learning and neural network architectures. Section 2.4 introduces graph theory, which provides the mathematical language for representing molecular structures as graphs. Section 2.5 describes the graph neural network architectures evaluated in this work in detail, covering both established and recently proposed models. Section 2.6 explains how molecular structures are represented as graphs and how atomic and bond features are encoded. Section 2.7 reviews related works in the literature. Section 2.8 concludes the chapter with a summary and directions for future research.

2.2 Machine Learning Fundamentals

Machine learning enables computational systems to learn patterns directly from data without being explicitly programmed with domain-specific rules. Unlike conventional approaches that required careful manual engineering of feature extractors to transform raw data into suitable representations, machine learning systems learn these representations automatically from labeled or unlabeled examples [10]. In the context of binding affinity prediction, machine learning methods offer a powerful framework for mapping molecular representations to continuous affinity values, enabling rapid screening of large compound libraries.

2.2.1 Supervised Learning

Supervised learning is the most widely used paradigm in machine learning, and forms the foundation of virtually all binding affinity prediction models. In supervised learning, a model is trained on a labeled dataset consisting of input–output pairs, where the goal is to learn a mapping function that generalizes well to unseen inputs. During training, the model produces predictions for each input and computes a loss function that measures the discrepancy between predictions and true labels. The model's internal parameters, commonly called weights, are then iteratively adjusted to minimize this loss [10].

The optimization process relies on computing the gradient of the loss function with respect to each weight, a quantity that indicates how much the loss would change if that weight were slightly modified. Weights are updated in the direction that reduces the loss, following the principle of gradient descent. In practice, most modern systems use stochastic gradient descent (SGD), which estimates the gradient using small random subsets of training examples called mini-batches rather than the full dataset. This introduces controlled noise into the optimization process, which can help escape poor local minima and improve generalization [10].

After training is complete, the model's performance is evaluated on a separate held-out test set that was not used during training. This evaluation measures the model's generalization ability and its capacity to produce accurate predictions for molecular systems it has never encountered before. In the context of this work, the model is trained on protein–ligand complexes from the PDBbind dataset and evaluated on a test split of unseen complexes.

2.2.2 Evaluation Metrics

To rigorously evaluate the performance of binding affinity prediction models, we employ four complementary evaluation metrics that capture different aspects of prediction quality. These metrics are computed between the predicted affinity values $\hat{y}_i$ and the ground truth values y_i over a test set of n samples.

Mean Absolute Error (MAE) measures the average magnitude of prediction errors without

penalizing large errors disproportionately [16]:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.1}$$

Lower MAE values indicate better model performance.

Root Mean Square Error (RMSE) quantifies the average magnitude of prediction errors while giving greater weight to larger errors, making it particularly sensitive to outliers [16]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.2}$$

Lower RMSE values indicate better model performance.

Pearson Correlation Coefficient (R) measures the strength of the linear relationship between predicted and true affinity values, where $\bar{y}_t$ and $\bar{y}_p$ denote the means of the true and predicted values respectively [16]:

$$R = \frac{\sum_{i=1}^{N}(y_t(i) - \bar{y}_t)(y_p(i) - \bar{y}_p)}{\sqrt{\sum_{i=1}^{N}(y_t(i) - \bar{y}_t)^2 \cdot \sum_{i=1}^{N}(y_p(i) - \bar{y}_p)^2}} \tag{2.3}$$

A larger R value indicates stronger agreement between predictions and ground truth.

Spearman Correlation Coefficient (ρ) measures the strength of the monotonic relationship between predicted and true affinity values, making it more robust than Pearson R when the relationship between predictions and true values is not strictly linear [16]:

$$\rho = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2 - 1)} \tag{2.4}$$

where $d_i = \text{rank}(y_t(i)) - \text{rank}(y_p(i))$ is the difference between the rank of the true value and the rank of the predicted value for sample i, and N is the number of samples. A larger ρ value indicates stronger monotonic agreement between predictions and ground truth.

2.3 Deep Learning and Neural Networks

Deep learning extends the machine learning paradigm by composing multiple layers of learnable transformations to build hierarchical representations of data. Rather than relying on manually engineered features, deep learning models learn to extract progressively more abstract representations directly from raw inputs through the repeated application of simple non-linear transformations [10]. This capacity to automatically discover useful representations from large datasets has driven remarkable advances in speech recognition, image clas-

sification, natural language processing, and more recently, computational drug discovery and genomics.

The defining characteristic of deep learning is that its internal representations are not designed by human engineers, they emerge from the training process itself, guided entirely by the data and the optimization objective [10]. In the context of molecular property prediction, this means that a deep learning model can learn to identify chemically meaningful patterns in atomic graphs or sequences without requiring explicit specification of which features are important.

2.3.1 Artificial Neural Networks

The fundamental building block of deep learning is the artificial neural network (ANN), which consists of a sequence of layers, each transforming its input through a learned linear transformation followed by a non-linear activation function. In a feedforward network, information flows in a single direction from the input layer through one or more hidden layers to the output layer, with no cycles or feedback connections [10].

Training a neural network requires computing the gradient of the loss function with respect to every learnable parameter in the network. This is accomplished efficiently through the backpropagation algorithm, which applies the chain rule of calculus to propagate error signals backwards from the output layer to the input layer. Starting from the loss at the output, gradients are computed layer by layer in reverse order, and the resulting gradients are used to update the weights via gradient descent [10]. Modern deep learning frameworks such as PyTorch implement automatic differentiation, making this process transparent and efficient even for networks with hundreds of millions of parameters.

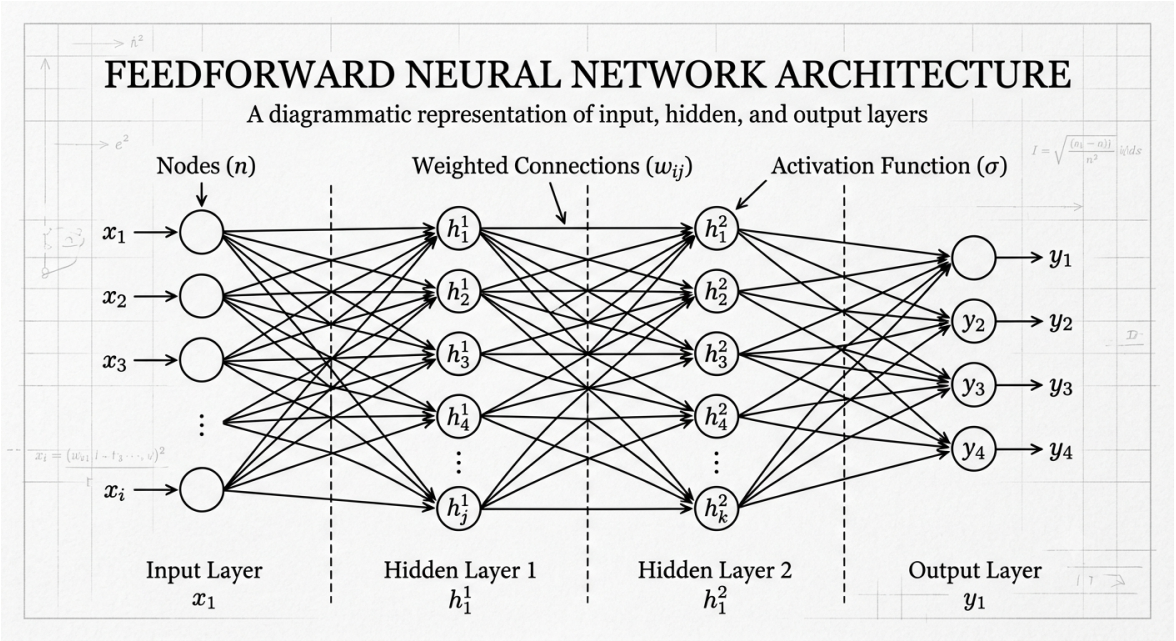


Figure 2.1: Artificial neural network diagram[10].

As the depth of a network increases, its capacity to represent complex functions grows substantially. A network with multiple non-linear layers can implement intricate input–output mappings that are simultaneously sensitive to fine-grained local details and robust to irrelevant global variations, making deep architectures particularly well-suited to the complexity of molecular binding interactions [10].

2.3.2 Activation Functions

Activation functions introduce the non-linearity that allows neural networks to approximate complex functions. Without them, stacking multiple linear layers would be equivalent to a single linear transformation, severely limiting the model's expressive power.

The **Rectified Linear Unit (ReLU)** is the most widely used activation function in modern deep learning due to its simplicity and its ability to mitigate the vanishing gradient problem that plagued earlier sigmoid-based networks [10]:

$$\text{ReLU}(z) = \max(0, z) \tag{2.5}$$

The **hyperbolic tangent (tanh)** is a smooth activation function that maps inputs to the range $(-1, 1)$ and was widely used in earlier neural network architectures:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.6}$$

The **sigmoid** (logistic) function maps inputs to the range $(0, 1)$ and is commonly used in

output layers for binary classification tasks:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

In this work, ReLU is used as the primary activation function throughout all GNN layers and fully connected layers, as it consistently produces stable training dynamics and strong empirical performance on regression tasks.

2.4 Graph Theory Fundamentals

The majority of data encountered in scientific applications cannot be represented on regular Euclidean grids such as images or sequences. Molecular structures, social networks, knowledge graphs, and protein interaction networks are all examples of data that are inherently irregular, their topology varies from sample to sample, and there is no natural notion of spatial translation invariance. Standard deep learning architectures such as convolutional neural networks were designed for grid-structured data and cannot be directly applied to such irregular domains [11]. Graph theory provides the mathematical language necessary to represent and reason about this type of data, and forms the foundation upon which graph neural networks are built.

2.4.1 Definition of a Graph

A graph $G = (V, E)$ consists of a set of nodes $V = \{v_1, \ldots, v_N\}$ representing entities and a set of edges E representing binary relations between them [11]. The adjacency matrix $A \in \mathbb{R}^{N \times N}$ encodes the connectivity structure of the graph, where $A_{ij} = 1$ if an edge exists between nodes v_i and v_j, and $A_{ij} = 0$ otherwise. Each node v_i is associated with a feature vector $x_i \in \mathbb{R}^d$, and the full matrix of node features is denoted $X = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times d}$ [12]. The neighborhood of a node u is defined as the set of all nodes connected to it by an edge:

$$\mathcal{N}(u) = \{v : (v, u) \in E\} \tag{2.8}$$

and the corresponding neighborhood feature multiset is $X_{\mathcal{N}(u)} = \{x_v : v \in \mathcal{N}(u)\}$ [12].

In the context of molecular graphs, nodes represent atoms and edges represent chemical bonds or spatial interactions between atoms. Node features encode atomic properties such as atom type, degree, formal charge, hybridization state, and aromaticity, while edge features encode bond properties such as bond type and bond length, as detailed in Section 2.6.

2.4.2 Graph Representations and the Need for Graph Neural Networks

Traditional deep learning architectures operate on data with a fixed and regular structure. Images are represented as two-dimensional grids where every pixel has the same number of neighbors, and text sequences have a fixed linear order. This regularity enables the use of shared convolutional filters that slide over the data in a translationally invariant manner [14].

Molecular graphs violate both of these properties. Different atoms have different numbers of neighbors, there is no canonical ordering of atoms in a molecule, and the local chemical environment of each atom is unique. As a result, the translation invariance that underlies standard convolution is no longer applicable, and a fundamentally different computational approach is required [14].

Graph neural networks address this challenge by defining a generalized convolution operation directly on the graph structure. Rather than applying a fixed filter at every spatial position, a graph neural network updates each node's representation by aggregating information from its neighbors, weighted by learnable parameters. The basic propagation rule of a graph convolutional network, introduced by Kipf and Welling, takes the following form [14]:

$$f(H^l, A) = \sigma\Big(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^l W^l\Big) \tag{2.9}$$

where $\hat{A} = A + I$ is the adjacency matrix augmented with self-loops, $\hat{D}$ is the corresponding diagonal degree matrix with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, H^l is the matrix of node representations at layer l, W^l is a learnable weight matrix, and $\sigma(\cdot)$ is a non-linear activation function. The normalization term $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ ensures that the aggregated features are scaled appropriately regardless of the degree of each node.

2.5 Graph Neural Networks (GNNs)

Graph neural networks were first introduced by Gori et al. and later formalized by Scarselli et al. as a general framework for learning on graph-structured data. In the original formulation, each node v maintains a state vector h_v that is updated iteratively by a shared transfer function f_w that takes as input the node's own features, its edge features, and the states of its neighbors [14]:

$$h_v = f_w\big(x_v,\, x_{co[v]},\, h_{ne[v]},\, x_{ne[v]}\big) \tag{2.10}$$

A local output function g_w then maps each node's state to an output representation:

$$o_v = g_w(h_v, x_v) \tag{2.11}$$

GNNs have since emerged as the dominant paradigm for molecular property prediction in drug discovery. Their ability to represent molecules as graphs where atoms serve as nodes and chemical bonds or spatial interactions serve as edges, allows them to capture both the topological structure and the chemical context of each atom in a way that sequence-based methods fundamentally cannot [17]. In this work, we evaluate thirteen distinct GNN architectures on the protein–ligand binding affinity prediction task, ranging from classical spectral methods to modern hybrid transformer-based models.

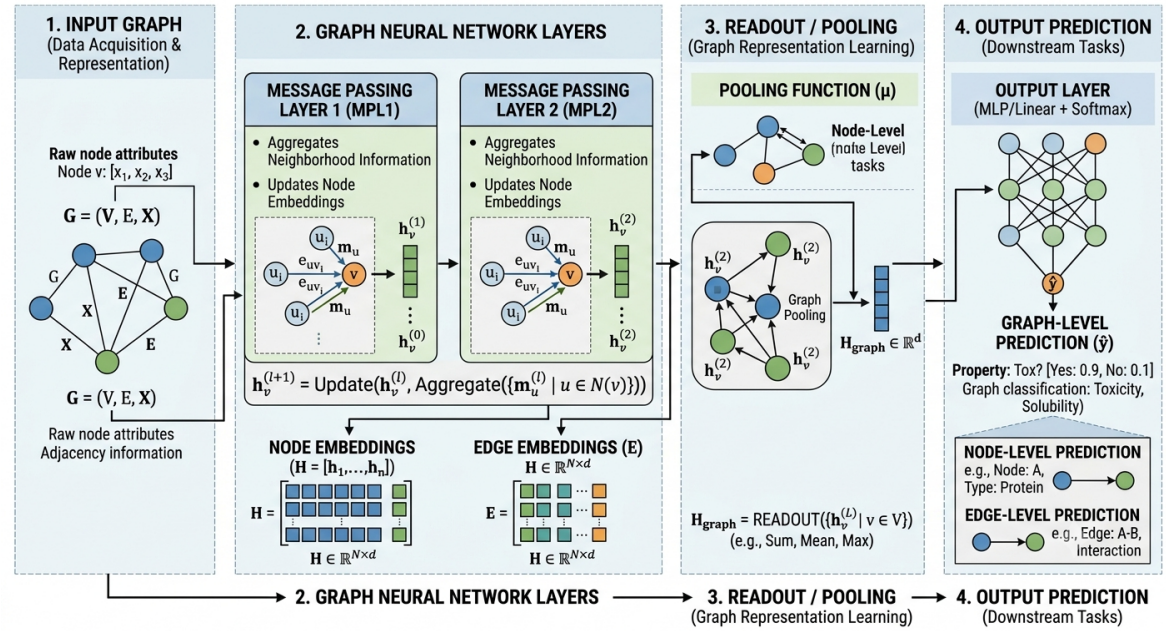


Figure 2.2: The end-to-end Graph Neural Network (GNN) pipeline for graph-level prediction [14].

2.5.1 Message Passing Framework

The unifying computational framework underlying the vast majority of modern GNNs is the message passing neural network (MPNN) formulation introduced by Gilmer et al. In this framework, every node iteratively updates its representation by aggregating messages from its neighbors and combining them with its own current representation [12]:

$$m_u^{(l)} = \text{Aggregate}^{(l)}\left\{h_v^{(l-1)} : v \in \mathcal{N}(u)\right\} \tag{2.12}$$

$$h_u^{(l)} = \text{Combine}^{(l)}\left(h_u^{(l-1)},\, m_u^{(l)}\right) \tag{2.13}$$

where $h_u^{(l)}$ denotes the hidden representation of node u at the l-th layer. The message passing process can also incorporate edge features e_{vw} through a message function M_l and a vertex

update function U_l [14]:

$$m_v^{(l+1)} = \sum_{w \in \mathcal{N}(v)} M_l\big(h_v^{(l)}, h_w^{(l)}, e_{vw}\big) \tag{2.14}$$

$$h_v^{(l+1)} = U_l\big(h_v^{(l)}, m_v^{(l+1)}\big) \tag{2.15}$$

After T layers of message passing, a graph-level representation is obtained by applying a differentiable readout function R over all node representations [14]:

$$\hat{y} = R\big(\{h_v^{(T)} \mid v \in G\}\big) \tag{2.16}$$

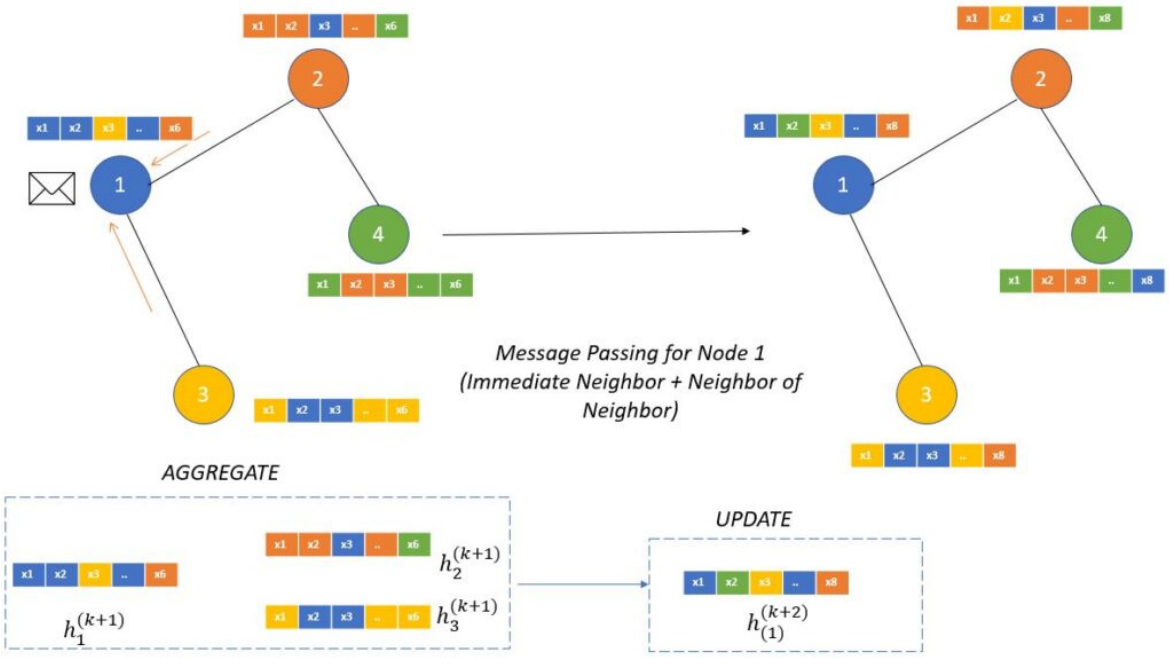


Figure 2.3: Graph Neural Network message passing aggregation update diagram[12].

By choosing different forms of the message function, aggregation function, update function, and readout function, the MPNN framework subsumes a wide variety of GNN architectures as special cases. The thirteen models evaluated in this work can all be expressed within this framework, differing primarily in how they define these components.

2.5.2 Graph Convolutional Networks (GCN)

Definition. Graph Convolutional Networks (GCN) were proposed by Kipf and Welling (2017) as a computationally efficient spectral graph convolution method based on a first-order approximation of localized spectral filters [14].

Architecture. GCN is grounded in spectral graph theory, where graph convolution is defined as the multiplication of a signal with a filter in the spectral domain of the graph Laplacian.

The spectral convolution of a graph signal x with a filter g_θ is given by [14]:

$$g_\theta \star x = U g_\theta(\Lambda) U^T x \tag{2.17}$$

where U is the matrix of eigenvectors of the normalized graph Laplacian and Λ is the diagonal matrix of its eigenvalues. To reduce the computational complexity of this operation, $g_\theta(\Lambda)$ is approximated using Chebyshev polynomials and truncated to first order, yielding the simplified layer-wise propagation rule [14]:

$$H^{(l+1)} = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{2.18}$$

where $\sigma(\cdot)$ is a non-linear activation function and $W^{(l)}$ is the learnable weight matrix of the l-th layer.

Key Advantage. Through hierarchical propagation, each node progressively aggregates information from an increasingly large neighborhood, effectively capturing multi-hop structural context with a simple and computationally efficient operation [13].

Justification. GCN was used in GraphDTA as one of the first GNN architectures applied to binding affinity prediction, establishing the value of graph-based molecular representations over sequence-based approaches [17].

2.5.3 Graph Attention Networks (GAT)

Definition. Graph Attention Networks (GAT) were proposed by Veličković et al. (2018) and extend GCN by replacing the fixed symmetric normalization with a learnable attention mechanism that assigns different importance weights to different neighbors [14].

Architecture. For each node i, GAT computes an attention coefficient a_{ij} that measures the importance of node j's features to node i. The attention coefficient is computed using a shared attention mechanism parametrized by a learnable vector α and weight matrix W [14]:

$$a_{ij} = \frac{\exp\big(\text{LeakyReLU}\big(\alpha^T[Wh_i \| Wh_j]\big)\big)}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\alpha^T[Wh_i\|Wh_k]))} \tag{2.19}$$

The updated node representation is then computed as a weighted sum of transformed neighbor features [14]:

$$h_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} a_{ij} W h_j\right) \tag{2.20}$$

To stabilize training and improve expressiveness, GAT employs multi-head attention, where K independent attention heads compute separate representations that are then concatenated or averaged.

Key Advantage. The attention mechanism allows GAT to selectively focus on the most relevant neighboring atoms, making it more expressive than GCN while remaining computationally tractable on molecular graphs of typical size.

Justification. In molecular graphs, not all neighboring atoms contribute equally to the binding interaction. GAT's ability to assign different weights to different atomic neighbors makes it well suited for learning which local chemical environments are most relevant to binding affinity prediction.

2.5.4 Graph Isomorphism Networks (GIN/GINE)

Definition. Graph Isomorphism Networks (GIN) were proposed by Xu et al. (2019) and are theoretically motivated by the Weisfeiler–Lehman (WL) graph isomorphism test, achieving the maximum discriminative power achievable by any MPNN [12].

Architecture. GIN uses a sum aggregation followed by a multi-layer perceptron (MLP) to update node representations. The graph-level representation is obtained by concatenating readout functions across all layers [16]:

$$h_G = \text{CONCAT}\big(f_r\big(\{h_v^k \mid v \in G\}\big)\big) \tag{2.21}$$

where h_v^k is the node representation at layer k, f_r is the readout function summing all node representations, and CONCAT combines outputs across layers. Each GIN layer includes batch normalization to improve training stability [16]:

$$\hat{h}_G = \gamma \times \frac{h_G - \bar{h}_G}{\sqrt{\delta(h_G) + \epsilon}} + (1 - \gamma) \tag{2.22}$$

where $\bar{h}_G$ is the mean, $\delta(h_G)$ is the standard deviation, γ is a learnable parameter, and ϵ is a small constant for numerical stability.

Key Advantage. By using sum aggregation rather than mean or max, GIN can distinguish between graphs that have different numbers of neighbors with the same features, a capability that mean and max aggregators lack.

Justification. GIN demonstrated the best performance among all GNN variants evaluated in the GraphDTA study on the KIBA and Davis datasets [17]. Its strong theoretical grounding and empirical track record make it one of the most reliable architectures for binding affinity prediction.

2.5.5 GraphSAGE

Definition. GraphSAGE (Graph SAmple and aggreGatE) was proposed by Hamilton, Ying, and Leskovec (2017) as an inductive framework for learning node representations on large graphs by sampling and aggregating features from local neighborhoods [12].

Architecture. Unlike transductive methods that require the full graph during training, GraphSAGE learns aggregator functions that can generate embeddings for previously unseen nodes. At each layer, a fixed-size sample of neighbors is drawn, and their features are aggregated using one of several aggregator variants. The mean aggregator computes an element-wise mean over neighbor features, while the LSTM aggregator feeds a random permutation of neighbor hidden states into an LSTM network [12]. The node representation is then updated by concatenating the node's current representation with the aggregated neighborhood representation and applying a learned linear transformation followed by a non-linear activation.

Key Advantage. GraphSAGE's inductive design allows it to generalize to molecular structures that were not present during training, which is essential for binding affinity prediction on compounds from new chemical scaffolds outside the training distribution.

Justification. The ability to generalize inductively to unseen protein–ligand complexes is particularly valuable in virtual screening scenarios, where the goal is to evaluate novel compounds against targets not represented in the training data.

2.5.6 TransformerConv

Definition. TransformerConv was proposed by Shi et al. (2021) within the Unified Message Passing (UniMP) framework, adapting the multi-head self-attention mechanism of the Transformer architecture to graph-structured data with explicit edge feature support [20].

Architecture. Given node features $H^{(l)} = \{h_1^{(l)}, \ldots, h_n^{(l)}\}$ at layer l, TransformerConv computes query and key vectors for each node pair connected by an edge [20]:

$$q_{c,i}^{(l)} = W_{c,q}^{(l)} h_i^{(l)} + b_{c,q}^{(l)} \tag{2.23}$$

$$k_{c,j}^{(l)} = W_{c,k}^{(l)} h_j^{(l)} + b_{c,k}^{(l)} \tag{2.24}$$

The attention weight for the c-th head on the edge from j to i incorporates edge features $e_{c,ij}$ directly into the key vector [20]:

$$\alpha_{c,ij}^{(l)} = \frac{\langle q_{c,i}^{(l)},\ k_{c,j}^{(l)} + e_{c,ij} \rangle}{\sum_{u \in \mathcal{N}(i)} \langle q_{c,i}^{(l)},\ k_{c,u}^{(l)} + e_{c,iu} \rangle} \tag{2.25}$$

where $\langle q, k \rangle = \exp(q^T k / \sqrt{d})$ is the scaled dot-product attention function. The aggregated

message for node i is computed as [20]:

$$\hat{h}_i^{(l+1)} = \|_{c=1}^{C} \left(\sum_{j \in \mathcal{N}(i)} \alpha_{c,ij}^{(l)} \left(v_{c,j}^{(l)} + e_{c,ij} \right) \right) \tag{2.26}$$

To prevent over-smoothing in deep networks, a gated residual connection is applied [20]:

$$\beta_i^{(l)} = \text{sigmoid}\Big(W_g^{(l)} \Big[\hat{h}_i^{(l+1)};\, r_i^{(l)};\, \hat{h}_i^{(l+1)} - r_i^{(l)} \Big] \Big) \tag{2.27}$$

$$h_i^{(l+1)} = \text{ReLU}\Big(\text{LayerNorm}\Big((1 - \beta_i^{(l)}) \hat{h}_i^{(l+1)} + \beta_i^{(l)} r_i^{(l)} \Big) \Big) \tag{2.28}$$

Key Advantage. By directly incorporating edge features into the attention computation, TransformerConv is able to leverage bond-level chemical information when determining how much each neighboring atom should contribute to the updated representation.

Justification. In our molecular graph construction, edges carry rich information including bond type, distance, and RBF-encoded interatomic distance. TransformerConv's native support for edge features makes it particularly well suited to exploit this information for binding affinity prediction.

2.5.7 Equivariant Graph Neural Networks (EGNN)

Definition. The E(n)-Equivariant Graph Neural Network (EGNN) was proposed by Satorras, Hoogeboom, and Welling (2021) as a computationally efficient architecture that explicitly respects the Euclidean symmetries of three-dimensional molecular systems [19].

Architecture. Given a graph $G = (V, E)$ where each node v_i has a feature vector $h_i^l \in \mathbb{R}^d$ and a 3D coordinate $x_i^l \in \mathbb{R}^3$, EGNN stacks E(n)-equivariant graph convolutional layers (EGCLs). Each layer computes edge messages using both node features and squared interatomic distances [19]:

$$m_{ij}^l = \phi_e\big(h_i^l,\, h_j^l,\, \|x_i^l - x_j^l\|^2,\, a_{ij} \big) \tag{2.29}$$

The 3D coordinates are updated equivariantly by accumulating direction-weighted messages from neighbors [19]:

$$x_i^{l+1} = x_i^l + \sum_{j \in \mathcal{N}(i)} \frac{(x_i^l - x_j^l)}{\|x_i^l - x_j^l\| + 1} \phi_x(m_{ij}^l) \tag{2.30}$$

The aggregated message for each node is computed as [19]:

$$m_i^l = \sum_{j \in \mathcal{N}(i)} m_{ij}^l \tag{2.31}$$

And the node features are updated as [19]:

$$h_i^{l+1} = \phi_h(h_i^l,\, m_i^l) \tag{2.32}$$

where ϕ_e, ϕ_x, and ϕ_h are multi-layer perceptrons. The resulting architecture is equivariant to rotations and translations of the coordinate system while remaining invariant in the feature space.

Key Advantage. EGNN achieves E(n)-equivariance without requiring expensive irreducible representations or Clebsch–Gordan decompositions, making it significantly more computationally efficient than other equivariant architectures while maintaining the ability to process 3D geometric information.

Justification. Protein–ligand binding is fundamentally a three-dimensional phenomenon, the relative positions and orientations of atoms in the binding pocket directly determine the strength of the interaction. EGNN's equivariance to rotations and translations ensures that its predictions are physically consistent regardless of how the complex is oriented in space.

2.5.8 GENeralized Aggregation Networks (GENConv)

Definition. GENeralized Aggregation Networks (GENConv) were proposed by Li et al. (2020) in the DeeperGCN paper, introducing differentiable generalized aggregation functions that overcome the vanishing gradient problem in very deep GCNs [21].

Architecture. The general message passing framework at layer l consists of three components message construction, message aggregation, and vertex update [21]:

$$m_{vu}^{(l)} = \rho^{(l)}\big(h_v^{(l)},\, h_u^{(l)},\, h_{e_{vu}}^{(l)}\big)\,, \quad u \in \mathcal{N}(v) \tag{2.33}$$

$$m_v^{(l)} = \zeta^{(l)}\big(\{m_{vu}^{(l)} \mid u \in \mathcal{N}(v)\}\big) \tag{2.34}$$

$$h_v^{(l+1)} = \phi^{(l)}\big(h_v^{(l)},\, m_v^{(l)}\big) \tag{2.35}$$

The key innovation of GENConv is to ensure all message features are positive using ReLU, enabling the use of generalized mean-max aggregations. The message construction function is defined as [21]:

$$m_{vu}^{(l)} = \mathrm{ReLU}\big(h_u^{(l)} + \mathbf{1}(h_{e_{vu}}^{(l)}) \cdot h_{e_{vu}}^{(l)}\big) + \epsilon \tag{2.36}$$

where $\mathbf{1}(\cdot)$ is an indicator function that is 1 when edge features exist and 0 otherwise, and $\epsilon = 10^{-7}$ is a small positive constant. Two families of generalized aggregation functions are proposed. The SoftMax aggregation interpolates between mean and max aggregation via an

inverse temperature parameter β [21]:

$$\text{SoftMax_Agg}_\beta(\cdot) = \sum_{u \in \mathcal{N}(v)} \frac{\exp(\beta m_{vu})}{\sum_{i \in \mathcal{N}(v)} \exp(\beta m_{vi})} \cdot m_{vu} \tag{2.37}$$

The PowerMean aggregation generalizes the arithmetic mean via a power parameter p [21]:

$$\text{PowerMean_Agg}_p(\cdot) = \left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} m_{vu}^p \right)^{1/p} \tag{2.38}$$

The vertex update function incorporates a Message Normalization (MsgNorm) layer that scales the aggregated message relative to the node's own feature norm [21]:

$$h_v^{(l+1)} = \text{MLP}\left(h_v^{(l)} + s \cdot \|h_v^{(l)}\|_2 \cdot \frac{m_v^{(l)}}{\|m_v^{(l)}\|_2} \right) \tag{2.39}$$

where s is a learnable scaling factor.

Key Advantage. GENConv's generalized aggregation functions can smoothly interpolate between mean and max aggregation during training, giving the model greater flexibility to adapt its neighborhood aggregation strategy to the specific characteristics of the molecular graph.

Justification. In protein–ligand complexes, different atoms may require fundamentally different aggregation strategies, some binding interactions are dominated by a single key atom (favoring max aggregation) while others depend on the collective contribution of many neighboring atoms (favoring mean aggregation). GENConv's learnable aggregation provides the flexibility to adapt to both scenarios.

2.5.9 Principal Neighbourhood Aggregation (PNA)

Definition. Principal Neighbourhood Aggregation (PNA) was proposed by Corso et al. (2020) and addresses the expressive power limitations of single-aggregator GNNs by combining multiple complementary aggregators with degree-aware scaling functions [22].

Architecture. PNA is motivated by a theoretical result showing that a single aggregation function is insufficient to distinguish all possible multisets of neighbor features over continuous domains. PNA leverages four aggregators mean, maximum, minimum, and standard deviation defined as [22]:

$$\mu_i(X^l) = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} X_j^l \tag{2.40}$$

$$\max_i(X^l) = \max_{j \in \mathcal{N}(i)} X_j^l, \qquad \min_i(X^l) = \min_{j \in \mathcal{N}(i)} X_j^l \tag{2.41}$$

$$\sigma_i(X^l) = \sqrt{\text{ReLU}(\mu_i(X^{l2}) - \mu_i(X^l)^2) + \epsilon} \tag{2.42}$$

In addition, PNA introduces degree-based scalers that amplify or attenuate messages according to the degree of the receiving node [22]:

$$S(d, \alpha) = \left(\frac{\log(d+1)}{\delta}\right)^{\alpha}, \quad d > 0, \quad -1 \leq \alpha \leq 1 \tag{2.43}$$

where δ is a normalization constant computed from the training set degree distribution and α controls whether the scaler amplifies ($\alpha > 0$) or attenuates ($\alpha < 0$) the messages. The full PNA aggregation combines four aggregators with three scalers (identity, amplification, attenuation) via the Kronecker product [22]:

$$M = \begin{bmatrix} I \\ S(D,\, \alpha = 1) \\ S(D,\, \alpha = -1) \end{bmatrix} \otimes \begin{bmatrix} \mu \\ \sigma \\ \max \\ \min \end{bmatrix} \tag{2.44}$$

The full PNA layer update is then [22]:

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(X_i^{(t)},\, E_{j\to i},\, X_j^{(t)}\right)\right) \tag{2.45}$$

where U is a learnable update function and $E_{j\to i}$ are optional edge features.

Key Advantage. By combining multiple aggregators and degree-aware scalers, PNA produces a richer and more informative neighborhood summary than any single aggregator, while the degree scalers correct for the systematic bias introduced by varying node degrees.

Justification. Molecular graphs exhibit significant variation in atom degree, carbon atoms in aromatic rings have very different connectivity patterns than nitrogen atoms at protein–ligand interfaces. PNA's degree-aware aggregation is designed precisely for such settings where local chemical environments vary significantly across the graph.

2.5.10 General, Powerful, Scalable Graph Transformer (GPS)

Definition. GPS (GraphGPS) was proposed by Rampášek et al. (2022) as a hybrid graph transformer architecture that combines local message passing with global attention mechanisms within a unified and scalable framework [23].

Architecture. The GPS recipe is built on three main ingredients: positional and structural encodings, a local message passing mechanism, and a global attention mechanism. At each layer l, node and edge features are updated by combining the outputs of an edge-aware MPNN

and a global attention module [23]:

$$\hat{X}_M^{l+1},\ E^{l+1} = \text{MPNN}_e^l\big(X^l,\ E^l,\ A\big) \tag{2.46}$$

$$\hat{X}_T^{l+1} = \text{GlobalAttn}^l\big(X^l\big) \tag{2.47}$$

$$X^{l+1} = \text{MLP}^l\Big(\hat{X}_M^{l+1} + \hat{X}_T^{l+1}\Big) \tag{2.48}$$

where $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix, $X^l \in \mathbb{R}^{N\times d^l}$ and $E^l \in \mathbb{R}^{|E|\times d^l}$ are the node and edge feature matrices at layer l, and MLP^l is a two-layer feed-forward network. Scalability is achieved through linear complexity $\mathcal{O}(N + |E|)$ in the number of nodes and edges [23]. Expressivity is enhanced through positional encodings (PE), which provide each node with a representation of its position within the graph, and structural encodings (SE), which capture local subgraph structure to improve the model's ability to distinguish non-isomorphic graphs [23].

Key Advantage. GPS overcomes the fundamental limitation of standard message-passing GNNs in their inability to capture long-range dependencies by combining local neighborhood aggregation with global self-attention, while maintaining linear computational complexity through the use of efficient attention approximations.

Justification. Protein–ligand binding affinity depends on both local atomic interactions at the binding interface and long-range electrostatic and hydrophobic effects that extend across the entire complex. GPS's hybrid architecture is uniquely positioned to capture both scales of interaction simultaneously.

2.5.11 Chebyshev Graph Convolution (ChebConv)

Definition. ChebConv was proposed by Defferrard, Bresson, and Vandergheynst (2016) as a spectral graph convolutional network that uses Chebyshev polynomial approximations to define fast and localized filters on graphs without requiring explicit computation of the graph Fourier basis [24].

Architecture. Graph convolution in the spectral domain is defined through the graph Laplacian $L = U\Lambda U^T$, where U is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues. The spectral filtering operation is [24]:

$$y = g_\theta(L)x = U g_\theta(\Lambda)U^T x \tag{2.49}$$

To avoid the computational cost of eigendecomposition and to ensure spatial localization of filters, the filter g_θ is parametrized as a truncated Chebyshev polynomial expansion of order

$K-1$ [24]:

$$g_\theta(\tilde{\Lambda}) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{2.50}$$

where $\theta \in \mathbb{R}^K$ is the vector of learnable Chebyshev coefficients, $\tilde{\Lambda} = 2\Lambda/\lambda_{\text{max}} - I_n$ is the rescaled eigenvalue matrix with values in $[-1, 1]$, and T_k is the Chebyshev polynomial of order k computed via the recurrence relation $T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L})$ with $T_0 = I$ and $T_1 = \tilde{L}$. The entire filtering operation costs $\mathcal{O}(K|E|)$ operations, making it significantly more efficient than full spectral methods [24].

Key Advantage. ChebConv's K-localized filters aggregate information from nodes within exactly K hops of each target node, providing precise control over the receptive field of each filter. Unlike GCN, which is limited to one-hop neighborhoods per layer, ChebConv can directly capture multi-hop chemical environments within a single layer.

Justification. In molecular binding, the chemical environment of an atom often extends beyond its immediate neighbors — the electronic effects of atoms two or three bonds away can significantly influence binding. ChebConv's multi-hop localized filters are designed to capture precisely this type of extended local chemical context.

2.5.12 Simple Graph Convolution (SGC)

Definition. Simple Graph Convolution (SGC) was proposed by Wu et al. (2019) and challenges the assumption that non-linearities between GCN layers are necessary, demonstrating that comparable performance can be achieved with a significantly simplified linear model [25].

Architecture. SGC successively removes the non-linear activation functions between GCN layers and collapses the stack of weight matrices into a single learned matrix. Starting from the standard GCN update rule, the repeated application of the normalized adjacency matrix $S = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ without intervening non-linearities yields [25]:

$$\hat{Y} = \text{softmax}(S^K X \Theta) \tag{2.51}$$

where S^K is the K-th power of the normalized adjacency matrix, X is the matrix of input node features, and Θ is a single learned weight matrix. The pre-computation of $S^K X$ is performed once before training and remains fixed, separating the feature smoothing step from the classification step. SGC can be interpreted as a fixed low-pass filter on the graph spectrum that produces smooth node features followed by a linear classifier [25].

Key Advantage. By pre-computing the graph diffusion $S^K X$ offline, SGC reduces inference to a single matrix multiplication, making it orders of magnitude faster than multi-layer GCNs while achieving comparable accuracy on many benchmark tasks. Its simplicity also reduces the risk of overfitting when training data is limited.

Justification. We include SGC as a computationally efficient baseline to quantify how much of the predictive performance of more complex GNN architectures can be attributed to the non-linear processing of molecular features, versus the simple smoothing of atomic features over the molecular graph structure.

2.6 Molecular Graphs

2.6.1 Representing Molecules as Graphs

In this work, molecular structures are represented as heterogeneous graphs that encode both the ligand and the protein binding pocket as a unified graph with three distinct types of nodes and edges. This representation allows GNN architectures to directly process the three-dimensional structure of the protein–ligand complex and learn the atomic-level interaction patterns that determine binding affinity.

Formally, a molecular graph is defined as $G = (V, E)$ where each node $v_i \in V$ represents an atom and each edge $(v_i, v_j) \in E$ represents either a covalent bond within the ligand, a covalent bond within the protein pocket, or a non-covalent spatial interaction between a ligand atom and a pocket atom. Ligand graphs $G^l = (N^l, M^l)$ are constructed from the ligand's three-dimensional SDF file, where M^l is the adjacency matrix encoding covalent bonds between atom pairs. Self-loops are added to the adjacency matrix, ensuring that each node incorporates its own features during convolution and is not solely influenced by its neighbors [5].

The protein binding pocket graph is constructed from the pocket PDB file by identifying all amino acid residues within a defined radius of the ligand. Interaction edges are then added between ligand atoms and pocket atoms whose Euclidean distance falls below a threshold of 5 Å, capturing the spatial proximity that drives non-covalent binding interactions. This construction results in a heterogeneous graph where three edge types are distinguished by an integer edge type flag: ligand bonds (type 0), pocket bonds (type 1), and interaction edges (type 2).

Molecular graphs have become the preferred representation for ligand encoding in binding affinity prediction due to their ability to capture both the topological connectivity and the three-dimensional spatial geometry of molecular structures. This format has been successfully employed in a wide range of deep learning models for drug discovery, including GraphDTA, DGraphDTA, DeepTGIN, and many others [17].

2.6.2 Node Features

Each atom in the molecular graph is represented by a 7-dimensional node feature vector encoding the most chemically informative atomic properties. These features were carefully

selected to provide the GNN with sufficient information to distinguish atoms by their chemical identity and bonding context, while keeping the feature dimension compact to reduce computational cost. The node features used in this work are summarized in Table 2.1.

Table 2.1: Node feature encoding used for all atoms in the molecular graph.

Feature	**Description**	**Encoding**
Atomic number	Chemical identity of the atom	`atomic_num / 100`
Degree	Number of covalent bonds	`degree / 5`
Formal charge	Net electric charge	`formal_charge / 5`
Aromaticity	Whether atom is in aromatic system	binary (0 or 1)
Hybridization	Orbital hybridization state	`hybridization / 10`
Partial charge	Gasteiger partial atomic charge	continuous scalar
Ligand flag	Whether atom belongs to ligand or pocket	binary (0 or 1)

All continuous features are normalized to the range $[0, 1]$ by dividing by their respective normalization constants. The ligand flag distinguishes ligand atoms from protein pocket atoms within the unified heterogeneous graph, enabling the model to learn separate representations for the two molecular components while still allowing information to flow across the protein–ligand interface through interaction edges.

2.6.3 Edge Features

Edge features encode the chemical and geometric properties of the interactions between connected atoms. The edge feature vector has a total dimension of 21, constructed from three components that vary according to the edge type.

For ligand bonds and pocket bonds (edge types 0 and 1), four base bond features are encoded: bond type (single, double, triple, aromatic), whether the bond is conjugated, whether the bond is part of a ring, and the bond stereo configuration. These features capture the chemical nature of covalent interactions within each molecular component.

For interaction edges (edge type 2), the normalized Euclidean distance between the connected atoms is encoded as a scalar, providing the model with direct geometric information about the spatial proximity of the ligand and pocket atoms. This distance feature is then expanded using a Radial Basis Function (RBF) encoding, as described in Section 2.6.4, to provide a smooth and differentiable representation of the interatomic distance.

In general, edge descriptors play a critical role in GNN-based binding affinity prediction because bond-level information including bond order, conjugation, and interatomic distance carries chemical information that cannot be inferred from node features alone. Edge features

allow the model to distinguish between fundamentally different types of atomic interactions and provide the geometric context necessary for accurate affinity prediction [7].

2.6.4 RBF Encoding of Interatomic Distances

Raw interatomic distances are scalar values that change continuously and may span a wide numerical range. Directly using a single scalar as an edge feature provides limited information to the GNN, as small differences in distance which may have large chemical significance are poorly represented. To address this, we expand each scalar interatomic distance d into a K-dimensional vector using Radial Basis Function (RBF) encoding, which provides a smooth and differentiable representation that captures distance information at multiple resolution levels simultaneously.

The RBF encoding is defined as a set of K Gaussian basis functions, each centered at a distinct reference distance μ_k and sharing a common width σ [18]:

$$\text{RBF}(d)_k = \exp\left(-\frac{(d-\mu_k)^2}{2\sigma^2}\right), \quad k = 1, \ldots, K \tag{2.52}$$

The centers μ_k are evenly spaced across the range of expected interatomic distances, and the width σ is set according to [18]:

$$\sigma = \frac{d_{\text{max}}}{\sqrt{2K}} \tag{2.53}$$

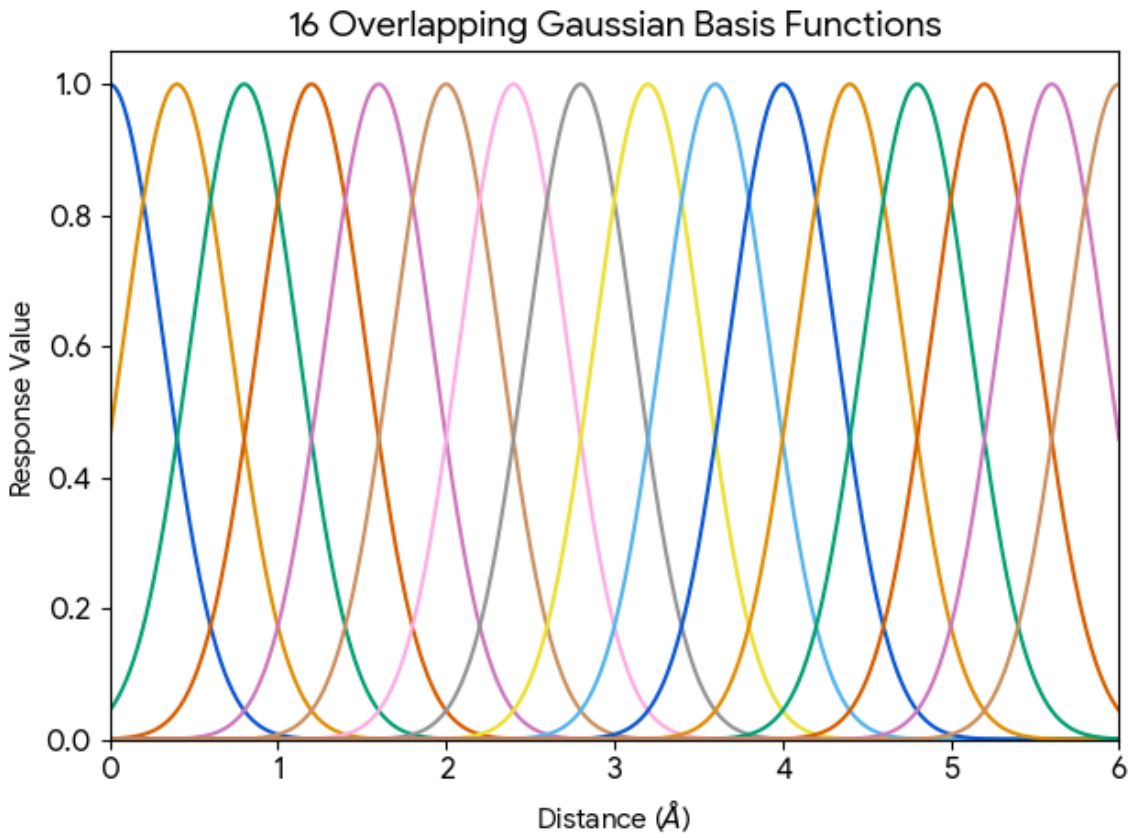


Figure 2.4: Sixteen Gaussian basis functions spanning the distance range 0–6 Å[18].

where d_{max} is the maximum interatomic distance considered and K is the number of basis functions. In this work, we use $K = 16$ Gaussian basis functions for interaction edges, yielding a 16-dimensional RBF vector. Combined with the normalized distance scalar, this produces a 17-dimensional distance encoding per interaction edge, which is then concatenated with the 4 base bond features to form the full 21-dimensional edge feature vector.

The RBF encoding has several important properties. Each basis function responds maximally when the interatomic distance equals its center μ_k and decays smoothly as the distance moves away from this value. The collective response of all K basis functions provides the model with a rich, localized encoding of the distance that is more informative than a single scalar and more compact than a one-hot discretization of the distance range.

2.7 Related Works

The prediction of protein–ligand binding affinity has been an active area of research for decades, with methods evolving from classical physics-based scoring functions to modern deep learning approaches. In recent years, there has been a clear paradigm shift toward graph neural network-based methods, driven by their ability to naturally represent molecular struc-

tures and capture complex atomic interaction patterns [17].

Early deep learning methods for binding affinity prediction, such as DeepDTA, represented both proteins and ligands as one-dimensional sequences amino acid sequences and SMILES strings respectively and applied convolutional neural networks to extract features from each. While these sequence-based methods demonstrated competitive performance on benchmark datasets such as Davis and KIBA, they fundamentally ignore the three-dimensional structural information that governs binding [17].

The introduction of GraphDTA by Nguyen et al. (2021) marked a turning point in the field by representing drug molecules as molecular graphs and applying GNNs to extract structural features. GraphDTA compared four GNN variants GCN, GAT, GIN, and a GCN-GAT combination for encoding the ligand graph, while retaining a CNN encoder for the protein sequence. The study demonstrated that graph-based ligand representations consistently outperform sequence-based SMILES representations across all GNN variants tested, with GIN achieving the best performance on both the KIBA and Davis datasets [17]. We got the same result where GINE outperform both GCN and GAT models.

PLAPT (Rose et al., 2024) represents a different approach by leveraging pre-trained transformer models for both the protein and ligand representations. Using ProtBERT for protein encoding and ChemBERTa for ligand encoding, PLAPT achieves state-of-the-art accuracy through a branching neural network architecture that integrates both embeddings for affinity prediction. Despite using minimal computational resources during fine-tuning, PLAPT surpasses numerous established baselines including DeepDTA, Pafnucy, OnionNet, and FAST on RMSE metrics [15].

DeepTGIN (Wang et al., 2024) introduces a hybrid multimodal architecture that combines the strengths of transformer-based sequence encoders and graph isomorphism networks. Transformers are used to extract contextual features from protein sequences and binding pocket representations, while GINs process the ligand molecular graph. Evaluated on both the PDBbind 2016 and PDBbind 2013 core sets, DeepTGIN outperforms state-of-the-art models across all reported metrics including R, RMSE, MAE, SD, and CI, achieving improvements of 4.3%, 9.1%, 10.7%, 8.4%, and 3.2% respectively over CAPLA [16].

Table 2.2 summarizes the performance of the most relevant related works on standard benchmark datasets.

Table 2.2: Summary of related works on GNN-based binding affinity prediction.

Reference	**Domain**	**Dataset**	**Model**	**RMSE**	R / R^2
[17]	Drug Discovery	KIBA	GCN, GIN, GAT	1.40	0.82
[17]	Drug Discovery	DAVIS	GIN	0.54	0.78
[16]	Drug Discovery	PDBbind 2016	GIN + Transformer	1.21	0.83
[15]	Drug Discovery	PDBbind	ProtBERT + ChemBERTa	1.31	0.79
[17]	Drug Discovery	PDBbind	DGraphDTA (GCN)	1.23	0.80
[17]	Drug Discovery	PDBbind	GraphDTA (GIN)	1.35	0.78
[16]	Drug Discovery	PDBbind 2013	DeepTGIN	1.58	0.78

Despite the significant progress achieved by these methods, several limitations remain. Most existing approaches either use sequence-based protein representations that discard structural information, or rely on structure-based representations that require high-quality crystal structures which may not always be available. Furthermore, the generalization of these models to novel protein families and chemically diverse ligands outside the training distribution remains a significant open challenge [17]. The present work addresses these limitations by constructing a unified heterogeneous graph that jointly represents the ligand and protein binding pocket in three-dimensional space, and by systematically comparing thirteen GNN architectures on this representation to identify which inductive biases are most beneficial for binding affinity prediction on the PDBbind dataset.

2.8 Conclusion

This chapter has presented the theoretical foundations underlying the methods employed in this work. We began with the core concepts of supervised machine learning and the evaluation metrics used to assess model performance, including MAE, RMSE, Pearson R, and Spearman R. We then introduced the principles of deep learning and the role of activation functions in enabling non-linear representations. Graph theory was introduced as the mathematical framework for representing molecular structures, motivating the need for graph neural networks over standard convolutional approaches.

The central contribution of this chapter is the systematic presentation of thirteen GNN architectures GCN, GAT, GIN, GraphSAGE, TransformerConv with two versions, EGNN, GENConv, PNA, GPS, ChebConv, and SGC each described in terms of its architecture, key advantage, and specific justification for binding affinity prediction. These models span a broad spectrum of design choices, from classical spectral methods to modern hybrid transformer-

based architectures, providing a comprehensive comparative framework for our experimental evaluation.

We then described how molecular structures are encoded as heterogeneous graphs in this work, detailing the node feature encoding, edge feature construction, and the RBF expansion of interatomic distances. Finally, the related works section situated our approach within the broader landscape of deep learning methods for binding affinity prediction, highlighting both the progress achieved by existing methods and the limitations that motivate our work.

The following chapter presents the methodology of our approach in detail, describing the complete graph construction pipeline, the base model architecture, and the experimental setup used to evaluate all thirteen GNN models on the PDBbind dataset.

Methodology

Chapter 3

Methodology

3.1 Introduction

This chapter presents the complete methodology employed in this work to predict protein–ligand binding affinity using graph neural networks. The approach follows a structured pipeline that begins with raw experimental data from the PDBbind database and concludes with the comparative evaluation of thirteen GNN architectures on a held-out test set. The pipeline consists of five main stages: dataset preprocessing and filtering, molecular graph construction, model architecture design, training configuration, and evaluation.

A central design decision in this work is the use of a unified and fixed base architecture that is shared across all evaluated models. Rather than designing a separate network for each GNN variant, we define a common framework consisting of a node embedding layer, four stacked GNN layers with residual connections, a dual pooling mechanism, and a shared regression head. The only component that varies across experiments is the type of graph convolution layer used. This controlled experimental design ensures that any differences in predictive performance between models can be attributed to the inductive biases of the GNN convolution operator, rather than to differences in network depth, pooling strategy, or output architecture.

The remainder of this chapter is structured as follows. Section 3.2 presents the project objectives. Section 3.3 provides a system architecture overview. Section 4.2.1 describes the technologies and tools used. Section 3.4 describes the dataset, its filtering procedure, and label statistics. Section 3.5 details the molecular graph construction pipeline. Section 3.6 explains label normalization and dataset splitting. Section 3.7 presents the shared base model architecture. Section 3.8 describes the training pipeline. Section 3.9 concludes the chapter.

3.2 Project Objectives

The primary objective of this work is to develop and comparatively evaluate a set of graph neural network architectures for the task of predicting protein–ligand binding affinity from three-dimensional molecular structures. Specifically, the project pursues the following goals:

1. Construct a unified molecular graph representation that jointly encodes the ligand and the protein binding pocket as a single heterogeneous graph, capturing covalent bonds within each component and non-covalent spatial interactions at the binding interface.
2. Implement a shared and reproducible training framework in which thirteen GNN architectures are evaluated under identical conditions, same dataset, same split, same loss function, same optimizer, and same base architecture, so that performance differences reflect only the choice of graph convolution operator.
3. Compare classical and modern GNN architectures ranging from simple spectral methods such as GCN and ChebConv to advanced hybrid models such as GPS and PNA, providing a comprehensive benchmark of GNN designs for structure-based binding affinity prediction.
4. Analyze model performance using four complementary metrics: RMSE, MAE, Pearson R, Spearman R. All metrics are computed in the original kcal/mol spaceto facilitate interpretation and comparison of the results.

3.3 System Architecture Overview

Figure 3.1 presents the overall pipeline of the proposed system. The pipeline consists of four sequential stages that transform raw PDBbind structural data into binding affinity predictions.

In the **Data Preparation** stage, the PDBbind dataset is loaded from disk, filtered to remove covalent, incomplete, and redundant entries, and the binding affinity labels are normalized using z-score standardization.

In the **Graph Construction** stage, each protein–ligand complex is converted into a heterogeneous molecular graph. The ligand structure is parsed from its SDF file and converted into a ligand subgraph. The protein binding pocket is parsed from its PDB file and converted into a pocket subgraph. The two subgraphs are then merged into a unified graph by adding bidirectional interaction edges between all ligand–pocket atom pairs within a 5 Å distance cutoff.

In the **Model Training** stage, the molecular graphs are batched and fed into the GNN model. Each model shares the same node embedding layer, four GNN layers with residual connections, dual pooling mechanism, and MLP regression head. Only the GNN convolution oper-

ator differs across the thirteen evaluated architectures. Training uses the AdamW optimizer with a ReduceLROnPlateau scheduler, gradient clipping, and early stopping.

In the **Evaluation** stage, the best checkpoint for each model is loaded and evaluated on the held-out test set. Predictions are de-normalized back to kcal/mol and four metrics are computed.

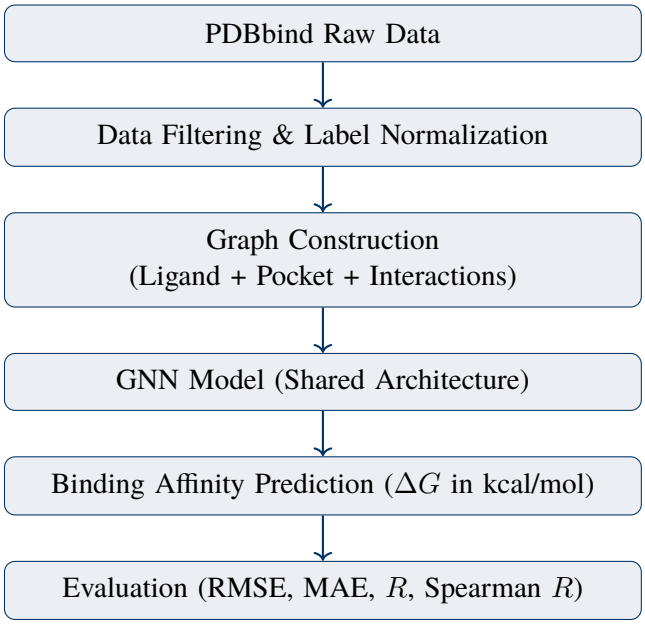


Figure 3.1: Overview of the proposed pipeline for protein–ligand binding affinity prediction.

3.4 Dataset

3.4.1 PDBbind Dataset

The PDBbind database is a curated subset of the Protein Data Bank (PDB) that collects experimentally determined three-dimensional structures of protein–ligand complexes together with their measured binding affinities, expressed as K_d, K_i, or IC_{50} values and converted to Gibbs free energy ΔG in kcal/mol[1]. It is the standard benchmark dataset for training and evaluating structure-based scoring functions for binding affinity prediction[17].

Each entry in the dataset provides two structural files: a ligand file in SDF format containing the three-dimensional coordinates of all ligand atoms, and a protein pocket file in PDB format containing the coordinates of all amino acid residues within the binding site[1]. These two files serve as the direct inputs to our graph construction pipeline. The dataset is organized into a general set and a refined set, where the refined set applies stricter quality criteria including crystallographic resolution, exclusion of covalent binders, and restriction to equilibrium binding constants[1] [1].

[1] http://www.pdbbind.org.cn

3.4.2 Data Filtering

Not all entries in the PDBbind dataset are suitable for machine learning. We apply three filtering criteria to ensure data quality. The filtering is performed by reading the labels CSV file and applying boolean conditions on three flags:

Listing 3.1: Data filtering applied to the PDBbind labels file.

```
df = df[
    (df["is_covalent"].astype(str).str.strip() == "False") &
    (df["is_incomplete"].astype(str).str.strip() == "False") &
    (df["is_redundant"].astype(str).str.strip() == "False")]
```

The three filters are applied as follows. First, entries flagged as covalent (`is_covalent = True`) are excluded because covalent binding involves the formation of a permanent chemical bond rather than the reversible non-covalent interaction that ΔG quantifies. Second, entries flagged as incomplete (`is_incomplete = True`) are excluded because missing structural information prevents reliable graph construction. Third, entries flagged as redundant (`is_redundant = True`) are excluded to reduce the risk of data leakage caused by highly similar protein–ligand pairs appearing in both training and test sets. After applying these three filters, the usable dataset contains approximately 17,366 protein–ligand complexes.

3.4.3 Label Distribution

The raw binding affinity labels are expressed in kcal/mol and represent the Gibbs free energy of binding ΔG. The distribution of ΔG values in the filtered dataset spans approximately from -20.76 to -0.54 kcal/mol, reflecting a wide range of binding strengths across different protein families and ligand chemotypes. The mean ΔG value is approximately -8.73 kcal/mol and the standard deviation is approximately 2.53 kcal/mol, indicating a roughly bell-shaped distribution centered around moderate binding affinities with a slight skew toward stronger binders. This distribution motivates the use of z-score normalization before training, as described in Section 3.6.

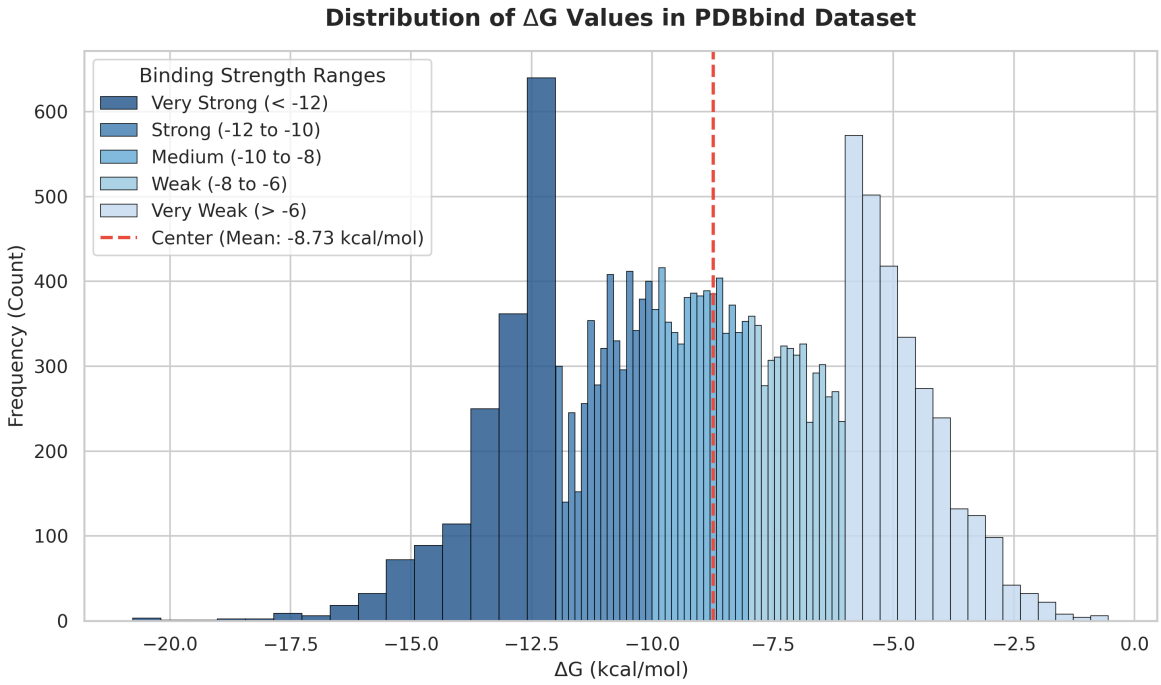


Figure 3.2: Histogram of binding free energy (ΔG) values in the filtered PDBbind dataset. The distribution is approximately Gaussian and centered near −8.73 kcal/mol.

3.5 Graph Construction

3.5.1 Ligand Graph

Each ligand is parsed from its SDF file using RDKit and converted into a molecular graph where each atom is a node and each covalent bond is an edge. Hydrogen atoms are removed during parsing since they contribute little discriminative information and would significantly increase graph size. Gasteiger partial charges are computed for each atom using RDKit's `ComputeGasteigerCharges` function.

Each atom is represented by a 7-dimensional node feature vector encoding the following properties, summarized in Table 3.1:

Table 3.1: Node feature encoding for ligand and pocket atoms.

Feature	**Encoding**	**Ligand flag value**
Atomic number	`atomic_num / 100`	1 (ligand) / 0 (pocket)
Degree	`degree / 5`	—
Formal charge	`formal_charge / 5`	—
Aromaticity	binary (0 or 1)	—
Hybridization	`hybridization / 10`	—
Partial charge	Gasteiger scalar	—
Ligand flag	binary (0 or 1)	1 for ligand atoms

Each covalent bond is encoded as a bidirectional edge. The edge feature vector for ligand bonds contains four base chemical features: bond type as a scalar (single$=1.0$, double$=2.0$, triple$=3.0$, aromatic$=1.5$), conjugation flag, ring membership flag, and an edge type identifier set to 0 to indicate a ligand bond. The RBF expansion of the interatomic distance is concatenated to form the full edge feature, as described in Section 3.5.3.

The following snippet illustrates the core of the ligand node feature extraction:

Listing 3.2: Node feature extraction for ligand atoms.

```
node_features.append([
    atomic_num / 100.0,
    degree / 5.0,
    formal_charge / 5.0,
    aromatic,
    hybridization / 10.0,
    partial_charge,
    is_ligand  # = 1 for ligand atoms
])
```

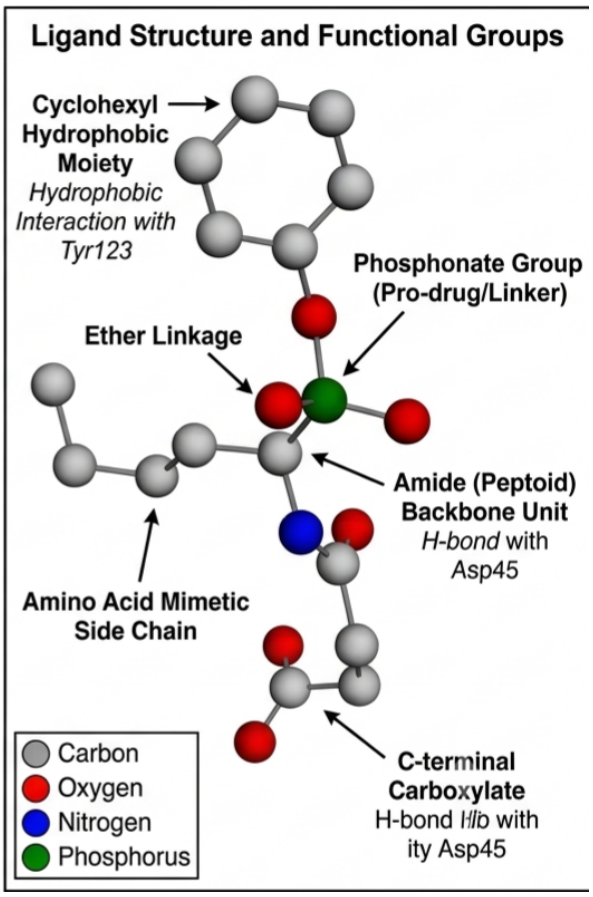


Figure 3.3: Three-dimensional ball-and-stick representation of a ligand molecule from the PDBbind dataset[1].

3.5.2 Pocket Graph

The protein binding pocket is parsed from its PDB file using RDKit in an analogous manner to the ligand. Each pocket atom is represented by the same 7-dimensional node feature vector as the ligand atoms, with the only difference being that the ligand flag is set to 0 to indicate a pocket atom. Covalent bonds within the pocket are encoded as bidirectional edges with the same four base chemical features as ligand bonds, but with the edge type identifier set to 1 to distinguish pocket bonds from ligand bonds.

Listing 3.3: Node feature extraction for pocket atoms.

```
node_features.append([
    atomic_num / 100.0,
    degree / 5.0,
    formal_charge / 5.0,
    aromatic,
    hybridization / 10.0,
    partial_charge,
    is_ligand  # = 0 for pocket atoms
```

])

The ligand flag therefore serves as a critical binary feature that allows the GNN to distinguish between the two molecular components within the unified heterogeneous graph, without requiring separate encoders for the ligand and the pocket.

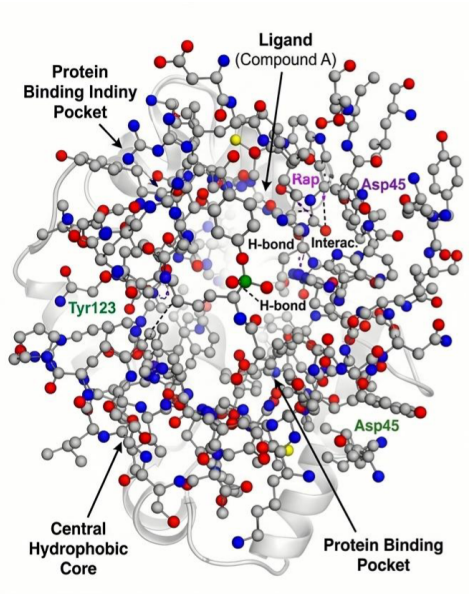


Figure 3.4: Three-dimensional ball-and-stick representation of the protein binding pocket extracted from the PDB file. Color coding: grey (Carbon), red (Oxygen), blue (Nitrogen), yellow (Sulfur)[1].

3.5.3 Interaction Edges

After constructing the ligand and pocket subgraphs separately, they are merged into a single heterogeneous graph. All pairwise Euclidean distances between ligand atoms and pocket atoms are computed using a vectorized distance matrix. All atom pairs whose distance falls below a cutoff of 5.0 Å are connected by bidirectional interaction edges:

Listing 3.4: Construction of interaction edges between ligand and pocket atoms.

```
dist_matrix = torch.cdist(ligand_pos, pocket_pos)
mask = dist_matrix <= cutoff
i_idx, j_idx = torch.where(mask)
```

Interaction edges capture the non-covalent contacts at the protein–ligand interface, the physical interactions that directly determine binding affinity. For each interaction edge, the normalized distance d/d_{cutoff} is encoded as a scalar feature, followed by a 16-dimensional Radial Basis Function (RBF) expansion. The RBF encoding transforms the scalar distance into a

smooth vector of Gaussian basis functions:

$$\mathrm{RBF}(d)_k = \exp\left(-\left(\frac{d-\mu_k}{\sigma}\right)^2\right), \quad k = 1, \ldots, 16 \tag{3.1}$$

where μ_k are 16 evenly spaced centers between 0 and 6 Å, and $\sigma = (d_{\max} - d_{\min})/K$ is the width of each basis function. The RBF encoding is implemented as:

Listing 3.5: RBF encoding of interatomic distances.

```
D_mu = torch.linspace(D_min, D_max, D_count)
D_sigma = (D_max - D_min) / D_count
return torch.exp(-((distances - D_mu) / D_sigma) ** 2)
```

The edge type identifier for interaction edges is set to 2. The complete interaction edge feature vector therefore has $4\,(\text{base}) + 1\,(\text{distance scalar}) + 16\,(\text{RBF}) = 21$ dimensions.

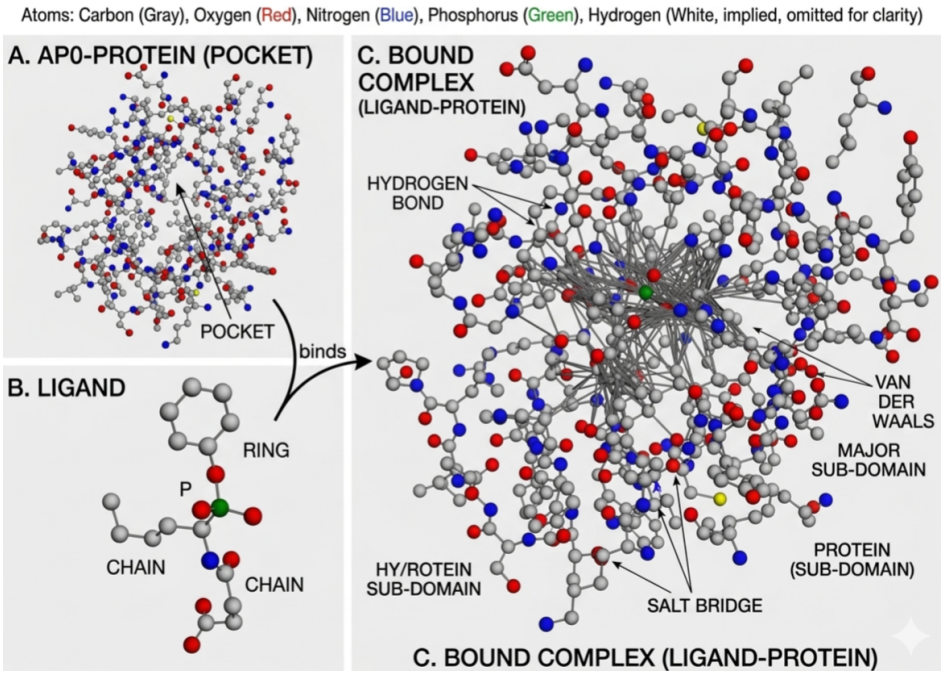


Figure 3.5: Visualization of the unified heterogeneous molecular graph for a protein-ligand complex from PDBbind[15].

3.5.4 Edge Feature Unification

The three edge types ligand bonds, pocket bonds, and interaction edges, are constructed independently and may have different feature dimensions before the RBF expansion is applied. To ensure that all edges in the merged graph have a consistent feature dimension equal to `EDGE_DIM = 21`, a padding function is applied to any edge feature tensor that is shorter than the target dimension:

Listing 3.6: Zero-padding to unify edge feature dimensions.

```
def pad_to(tensor, target):
    diff = target - tensor.size(1)
    if diff > 0:
        tensor = torch.cat([tensor,
            torch.zeros(tensor.size(0), diff)], dim=1)
    return tensor

lig_attr = pad_to(ligand.edge_attr, EDGE_DIM)
poc_attr = pad_to(pocket.edge_attr, EDGE_DIM)
```

This zero-padding ensures that the GNN's edge feature projection layer receives a consistent input dimension regardless of edge type. The three edge attribute tensors — padded ligand bonds, padded pocket bonds, and interaction edges — are then concatenated along the edge dimension to form the final unified edge attribute matrix for the merged graph. Table 3.2 summarizes the composition of the final 21-dimensional edge feature vector.

Table 3.2: Composition of the unified 21-dimensional edge feature vector.

Component	**Dim.**	**Applies to**
Bond type (scalar)	1	All edges
Conjugation flag	1	All edges
Ring membership flag	1	All edges
Edge type identifier (0, 1, 2)	1	All edges
Normalized distance scalar	1	Interaction edges only
RBF expansion (16 Gaussians)	16	Interaction edges only
Total	**21**	

3.6 Dataset Class and Label Processing

3.6.1 Label Normalization

The raw binding affinity labels span a wide numerical range (approximately -20.76 to -0.54 kcal/mol), which can cause unstable gradient magnitudes during training. We apply z-score normalization to all labels before training:

$$\tilde{y} = \frac{y - \mu}{\sigma} \tag{3.2}$$

where y is the raw ΔG value in kcal/mol, μ is the mean and σ is the standard deviation of all ΔG values in the filtered dataset. The normalization statistics are computed from the full filtered dataset and stored in the dataset class. During evaluation and testing, all predictions are de-normalized back to kcal/mol using the inverse transformation:

$$y = \tilde{y} \cdot \sigma + \mu \tag{3.3}$$

The de-normalization is applied through a `denormalize` method in the dataset class before any metric computation, ensuring that all reported results RMSE, MAE, Pearson R, and Spearman R are expressed in the original kcal/mol scale and are directly comparable with results from the literature.

3.6.2 Train/Validation/Test Split

The filtered dataset is partitioned into three non-overlapping subsets using stratified random splitting with a fixed random seed of 42 to ensure full reproducibility. The split is performed in two steps. First, 20% of the dataset is held out as the test set. Then, 10% of the remaining data is held out as the validation set. The resulting split ratios are approximately 72% training, 8% validation, and 20% testing.

Listing 3.7: Dataset splitting with fixed random seed.

```
train_idx, test_idx = train_test_split(
    indices, test_size=0.2, random_state=42)
train_idx, val_idx = train_test_split(
    train_idx, test_size=0.1, random_state=42)
```

The training set is used exclusively to update model parameters. The validation set is used to monitor convergence and trigger early stopping. The test set is held out entirely until the final evaluation and is never used during model selection or hyperparameter tuning.

3.7 Model Architecture

All thirteen GNN models evaluated in this work share an identical overall architecture. The only component that differs across models is the type of graph convolution operator applied in the four GNN layers. This section describes each shared component.

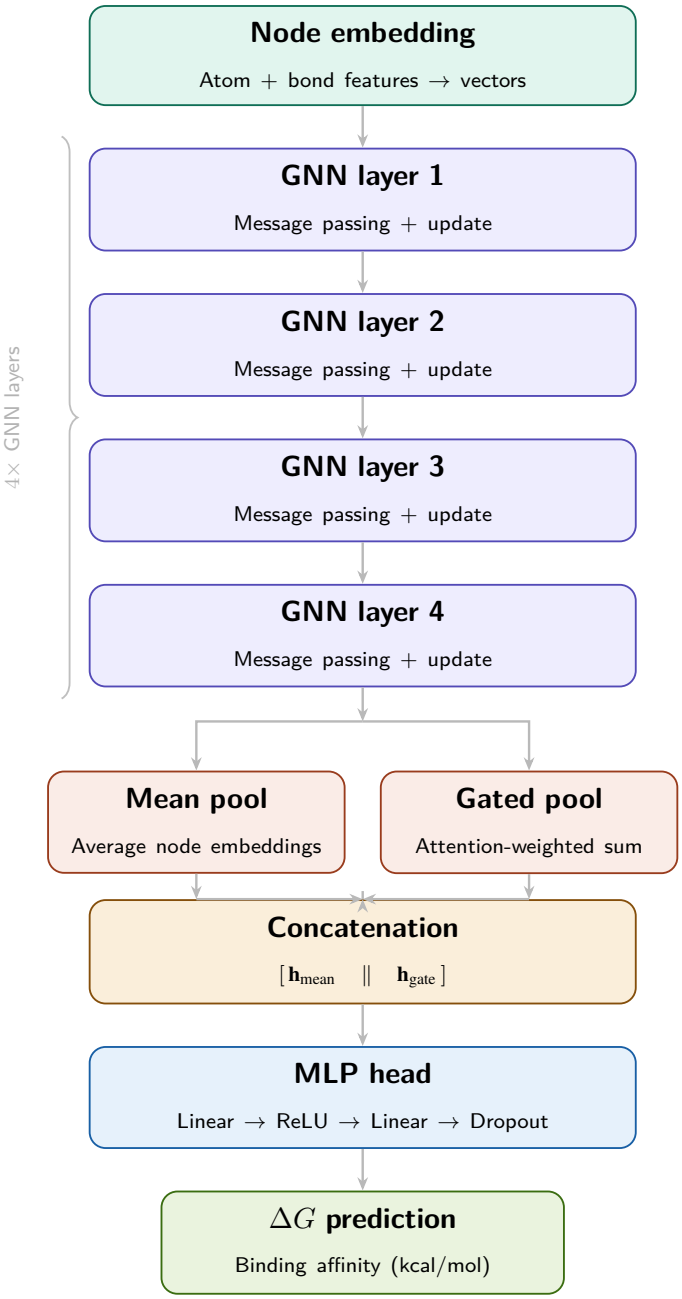


Figure 3.6: Base GNN architecture for binding affinity prediction.

3.7.1 Input Projection

The raw 7-dimensional node feature vectors are first projected into a 128-dimensional hidden space using a learnable linear embedding layer:

$$h_i^{(0)} = W_{emb} \cdot x_i + b_{emb} \tag{3.4}$$

where $x_i \in \mathbb{R}^7$ is the raw feature vector of atom i and $h_i^{(0)} \in \mathbb{R}^{128}$ is the initial node embedding. This projection maps all atoms whether ligand or pocket into the same representation space, enabling consistent message passing across the entire heterogeneous graph.

For models that require edge features in the hidden dimension (GENConv and PNA), an additional linear edge embedding layer $W_{edge} \in \mathbb{R}^{128\times 21}$ projects the 21-dimensional edge features into the 128-dimensional hidden space before the first GNN layer.

3.7.2 GNN Layers

The core of the architecture consists of four stacked GNN layers. Each layer updates node representations by aggregating information from neighboring atoms. Residual connections are applied from layer 2 onward to facilitate gradient flow and prevent over-smoothing:

$$h^{(1)} = \text{GELU}\big(\text{CONV}_1(h^{(0)}, \mathcal{E})\big) \tag{3.5}$$

$$h^{(l)} = \text{GELU}\big(\text{CONV}_l(h^{(l-1)}, \mathcal{E})\big) + h^{(l-1)}, \quad l = 2, 3, 4 \tag{3.6}$$

where $\mathcal{E}$ denotes the edge index and edge attribute tensors, and CONV_l is the graph convolution operator at layer l. The GELU activation function is used throughout as it provides smoother gradients than ReLU for regression tasks. The thirteen operators evaluated for CONV_l are listed in Table 3.3.

Table 3.3: GNN operators evaluated in this work.

Model	**Operator**	**Edge Feats**	**Attention**
GCN	`GCNConv`	No	No
GAT	`GATv2Conv`	Yes	Yes
GINE	`GINEConv`	Yes	No
GraphSAGE	`SAGEConv`	No	No
TransformerConv	`TransformerConv`	Yes	Yes
TransformerConv v2	`TransformerConv + BN`	Yes	Yes
GENConv	`GENConv`	Yes (projected)	No
PNA	`PNAConv`	Yes (projected)	No
GPS	`GPSConv + GINEConv`	Yes	Yes
Mixed	`GINE+Trans+GEN+GAT`	Partial	Partial
ChebConv	`ChebConv` ($K = 3$)	No	No
SimpleConv	`SimpleConv`	No	No
EGNN	`EGCL`	yes	No

3.7.3 Global Pooling

After four layers of message passing, each atom has a 128-dimensional representation. To obtain a single fixed-size graph-level representation, we apply a dual pooling mechanism

combining two complementary strategies.

The first component is **global mean pooling**, which computes the element-wise mean over all node representations:

$$h_{mean} = \frac{1}{|V|} \sum_{i \in V} h_i^{(4)} \tag{3.7}$$

The second component is **gated attention pooling**, where each node is assigned a scalar gate score and the pooled representation is a weighted sum of transformed node features:

$$\alpha_i = \text{sigmoid}(W_{gate} \cdot h_i^{(4)}), \qquad h_{attn} = \sum_{i \in V} \alpha_i \cdot (W_{tr} \cdot h_i^{(4)}) \tag{3.8}$$

The two pooled vectors are concatenated to produce the final 256-dimensional graph representation:

$$h_G = [h_{mean} \| h_{attn}] \in \mathbb{R}^{256} \tag{3.9}$$

3.7.4 MLP Head

The 256-dimensional graph representation is passed through a three-layer MLP regression head to produce the final scalar binding affinity prediction:

$$\hat{y} = W_3 \cdot \text{GELU}(W_2 \cdot \text{Dropout}(\text{GELU}(W_1 \cdot h_G))) \tag{3.10}$$

where $W_1 \in \mathbb{R}^{128 \times 256}$, $W_2 \in \mathbb{R}^{32 \times 128}$, $W_3 \in \mathbb{R}^{1 \times 32}$, and dropout with probability 0.1 is applied after the first linear layer. The output $\hat{y}$ is a scalar in the normalized label space, which is de-normalized to kcal/mol using equation (3.3) during evaluation.

3.8 Training Pipeline

3.8.1 Loss Function

We use a custom `RangeAwareLoss` that combines two complementary objectives. The primary term is the Huber loss, which provides robustness to outlier affinities while maintaining differentiable gradients near zero error:

$$\mathcal{L}_{Huber}(\hat{y}, y) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2 & \text{if } |\hat{y} - y| \leq \delta \\ \delta \cdot \left(|\hat{y} - y| - \frac{\delta}{2} \right) & \text{otherwise} \end{cases} \tag{3.11}$$

with $\delta = 1.5$. The secondary term is a range penalty that penalizes the model when its predicted standard deviation within a batch falls below the target standard deviation:

$$\mathcal{L}_{range} = \text{ReLU}(\sigma_y - \sigma_{\hat{y}}) \tag{3.12}$$

This penalty prevents the common failure mode of regression collapse, where the model learns to predict near the mean of the training distribution instead of spreading its predictions across the full affinity range. The total loss is:

$$\mathcal{L} = \mathcal{L}_{Huber} + \lambda \cdot \mathcal{L}_{range} \tag{3.13}$$

where $\lambda = 0.3$ is the range penalty weight.

3.8.2 Optimizer and Scheduler

All models are trained using the AdamW optimizer with a learning rate of 2×10^{-4} and weight decay of 10^{-5}. AdamW decouples weight decay regularization from the gradient update, providing more reliable regularization for deep networks compared to standard Adam.

A `ReduceLROnPlateau` scheduler monitors the negative validation Pearson R and reduces the learning rate by a factor of 0.7 if no improvement is observed for 2 consecutive epochs. Gradient clipping with a maximum norm of 5.0 is applied before each parameter update to prevent exploding gradients:

$$g \leftarrow g \cdot \min\left(1, \frac{5.0}{\|g\|}\right) \tag{3.14}$$

Training is stopped early if the validation Pearson R does not improve for 15 consecutive epochs. The maximum number of training epochs is 200, and the batch size is 32 graphs per mini-batch.

3.8.3 Checkpointing

After each epoch, a complete checkpoint is saved containing the model state, optimizer state, scheduler state, current epoch number, best validation and test metric, and all training history arrays. This allows training to be resumed exactly from any previous epoch without losing any progress:

Listing 3.8: Saving a training checkpoint after each epoch.

```
torch.save({
    'epoch': epoch,
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'best_val_loss': best_val_loss,
```

```
    'train_losses': train_losses,
    'val_pearsons': val_pearsons,
    'test_rmses': test_rmses,
    'test_maes': test_maes,
    'test_pearsons': test_pearsons,
    'test_spearmans': test_spearmans
}, checkpoint_path)
```

A separate best model file is saved whenever the validation Pearson R improves. This best model checkpoint is the one loaded for final evaluation on the test set. The separation between the rolling checkpoint (used for resuming) and the best model checkpoint (used for evaluation) ensures that training interruptions on Kaggle (caused by session timeouts) do not affect the quality of the final results.

3.9 Conclusion

This chapter has presented the complete methodology employed in this work. We described the project objectives and the overall system architecture as a four-stage pipeline. We presented the six tools used Python, PyTorch, PyTorch Geometric, RDKit, Google Colab, and Kaggle with their roles and justifications. We described the PDBbind dataset, the three-filter data cleaning procedure, and the ΔG label distribution.

We then detailed the molecular graph construction pipeline, which represents each protein–ligand complex as a unified heterogeneous graph with 7-dimensional node features and 21-dimensional edge features. The graph encodes covalent bonds within the ligand (edge type 0), covalent bonds within the pocket (edge type 1), and non-covalent interaction edges within a 5 Å cutoff (edge type 2), with the interaction edges enriched by RBF-encoded distances.

We presented the shared base architecture common to all thirteen models comprising a linear input projection, four GNN layers with residual connections, dual mean and gated pooling, and a three-layer MLP regression head. We described the `RangeAwareLoss` combining Huber loss with a range penalty, and the training configuration using AdamW, `ReduceLROnPlateau`, gradient clipping, early stopping, and checkpoint-based resumption.

The following chapter presents the experimental results of evaluating all thirteen GNN models on the PDBbind test set, analyzes their comparative performance across the four evaluation metrics, and discusses the implications of the observed results.

Experimental Results and Discussion

Chapter 4

Experimental Results and Discussion

4.1 Introduction

This chapter presents the experimental results obtained from training and evaluating thirteen Graph Neural Network architectures on the protein-ligand binding affinity prediction task. The models were trained under identical conditions using the PDBbind dataset, and their performance was evaluated using four standard regression metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Pearson correlation, and Spearman rank correlation.

We begin by describing the development environment, followed by presenting the experimental setup including the training parameters and evaluation protocol. Then a presentation of model's results presenting performance comparison table across all models. providing a detailed analysis of the best-performing model. The chapter concludes with a a comparison with existing works from literature and discussion of the factors influencing model performance and the challenges, obstacles encountered during this work. ends with outline directions for future research.

4.2 Development Environment

4.2.1 Technologies and Tools

4.2.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics, designed to emphasize readability and rapid application development [26]. We chose Python as the primary implementation language for this project because of its dominant position in the scientific computing and deep learning ecosystems, and because all major libraries used in this work: PyTorch, PyTorch Geometric, RDKit, and NumPy provide native

Python interfaces.

4.2.3 PyTorch

PyTorch is an open-source machine learning library that provides an imperative and Pythonic programming style, supporting dynamic computation graphs, automatic differentiation, and GPU acceleration [27]. We chose PyTorch because it is the standard framework for graph neural network research, offering native integration with PyTorch Geometric and providing flexible model definition that allowed us to implement a shared base architecture across all thirteen GNN variants with minimal code duplication.

4.2.4 PyTorch Geometric

PyTorch Geometric (PyG) is a deep learning library built on top of PyTorch for representation learning on graphs and other irregular data structures, achieving high throughput through sparse GPU kernels and an efficient message passing API [28]. We chose PyG because it provides production-ready implementations of all the GNN convolution operators evaluated in this work including `GCNConv`, `GATv2Conv`, `GINEConv`, `TransformerConv`, `PNAConv`, `GPSConv`, `GENConv`, `EGNN` `ChebConv`, and `SimpleConv` under a unified interface, eliminating the need to implement each architecture from scratch and ensuring that all models are compared on identical, well-tested implementations.

4.2.5 RDKit

RDKit is a widely used open-source cheminformatics toolkit that provides comprehensive tools for reading, writing, processing, and analyzing chemical structures [29]. We use RDKit in this work to parse ligand SDF files and protein PDB files, to extract atomic properties such as atom type, degree, formal charge, hybridization state, and aromaticity, and to compute Gasteiger partial charges. RDKit is the standard choice for molecular graph construction in deep learning for drug discovery, and its Python API integrates naturally with the PyTorch Geometric data pipeline.

4.2.6 Google Colab

Google Colaboratory (Colab) is a free cloud-based notebook environment provided by Google that offers access to GPU and TPU hardware for running computationally intensive code [30]. We used Colab during the early development and debugging stages of this project, particularly for initial graph construction and model prototyping, taking advantage of its free GPU access and its seamless integration with Google Drive for dataset storage.

4.2.7 Kaggle

Kaggle is a data science platform that provides free access to GPU computing resources, public datasets, and collaborative notebook environments [31]. We used Kaggle as the primary training environment for this project, running all model training experiments on Kaggle's T4 GPU instances. Kaggle's persistent storage and notebook versioning allowed us to track training runs, save checkpoints reliably, and resume training across sessions which was essential given the scale of training thirteen GNN models on the full PDBbind dataset.

4.3 Experimental Setup

4.3.1 Training Parameters

All thirteen models were trained under identical hyperparameter configurations to ensure a fair and controlled comparison. The training parameters are summarized in Table 4.1.

Table 4.1: Training hyperparameters applied uniformly to all evaluated models.

Parameter	**Value**	**Rationale**
Hidden dimension	128	Balances model capacity with memory constraints on free GPU instances
GNN layers	4	Consistent depth across all models for fair comparison
Attention heads	4	Standard configuration for attention-based GNN models
Learning rate	2×10^{-4}	Provides stable convergence across all GNN architectures without requiring per-model tuning
Weight decay	1×10^{-5}	Light regularization to prevent overfitting
Batch size	32	Largest batch size that can fit within GPU memory for all model variants
Max epochs	200	Most models converge earlier via early stopping but some does not

Continued on next page

Parameter	Value	Rationale
Optimizer	AdamW	Standard optimizer for GNN tasks with weight decay from gradient updates, providing more reliable regularization than Adam
Loss function	RangeAwareLoss	Combines Huber loss (δ=1.5) with range penalty (λ=0.3) to prevent prediction collapse
Scheduler	ReduceLROnPlateau	Adaptive learning rate reduction based on validation R that shows no improvement for 2 epochs
Early stopping patience	15	Stops training if validation Pearson R does not improve for 15 consecutive epochs
Gradient clipping	5.0	Prevents exploding gradients in deep GNN architectures
Dropout	0.1	Applied in the MLP regression head to reduce overfitting
Random seed	42	Fixed for full reproducibility of data splits and model initialization
Node features	7	Atomic number, degree, formal charge, aromaticity, hybridization, partial charge, ligand flag
Edge features	21	4 bond features + 1 distance scalar + 16 RBF-encoded distances
Interaction cutoff	5.0 Å	Standard threshold for capturing non-covalent protein-ligand contacts

4.3.2 Evaluation Protocol

Model performance is evaluated on the independent test set, which contains 20% of the filtered PDBbind dataset and was never used during training or model selection. For each model, the checkpoint corresponding to the epoch with the highest validation Pearson R is loaded for final test evaluation. All predictions are de-normalized from the z-score space back to the original kcal/mol scale before computing any metric, ensuring that all reported values

are directly interpretable and comparable with results from the literature.

We report four complementary evaluation metrics for each model. The Test RMSE (Root Mean Square Error) measures the average magnitude of prediction errors with greater sensitivity to large outliers. The Test MAE (Mean Absolute Error) measures the average absolute prediction error with equal weighting across all samples. The Test Pearson R measures the strength of the linear relationship between predicted and true affinity values. The Test Spearman R measures the rank-order correlation between predictions and true values, which is particularly relevant for virtual screening applications where correct compound ranking matters as much as absolute accuracy.

4.4 Performance Comparison Table

Table 4.2 presents the complete performance comparison of all thirteen evaluated GNN architectures on the PDBbind test set, sorted by test Pearson correlation R in descending order. Where ↓ = lower is better, ↑ = higher is better.

Table 4.2: Performance comparison of all evaluated GNN models on the PDBbind test set. Models are sorted by test Pearson correlation R (descending).

Rank	Model	Epochs	Train Loss	Val RMSE↓	Val R↑	Val Sp.↑	Test RMSE↓	Test MAE↓	Test R↑	Test Sp.↑
1	GENConv	45	0.2376	1.7508	0.7326	0.7314	1.7689	1.3580	0.7209	0.7152
2	GPS	44	0.2974	1.8200	0.7075	0.7055	1.7641	1.3741	0.7197	0.7116
3	TransformerConv v2	68	0.3519	1.8536	0.7244	0.7226	1.8239	1.4367	0.7138	0.7071
4	Mixed	50	0.3621	1.8699	0.7000	0.7004	1.8113	1.3977	0.7090	0.7012
5	ChebConv	70	0.3118	1.8357	0.7006	0.7003	1.8033	1.3970	0.7051	0.6928
6	TransformerConv v1	89	0.3985	1.8360	0.6978	0.6912	1.8001	1.4048	0.7039	0.6946
7	GINE	58	0.2872	1.8517	0.7032	0.7085	1.8501	1.3580	0.6974	0.6940
8	EGNN	60	0.3692	1.8880	0.6800	0.6748	1.8372	1.4267	0.6912	0.6821
9	PNA	39	0.2918	1.8514	0.6936	0.6959	1.8497	1.4752	0.6870	0.6785
10	GraphSAGE	89	0.4925	1.9719	0.6437	0.6441	1.9012	1.4752	0.6599	0.6518
11	GAT	126	0.5374	2.0220	0.6320	0.6337	1.9256	1.4996	0.6589	0.6523
12	GCN	151	0.5760	2.0590	0.5997	0.5939	1.9571	1.5346	0.6329	0.6178
13	SimpleConv	57	0.6424	2.1291	0.5572	0.5525	2.0250	1.5897	0.5959	0.5806

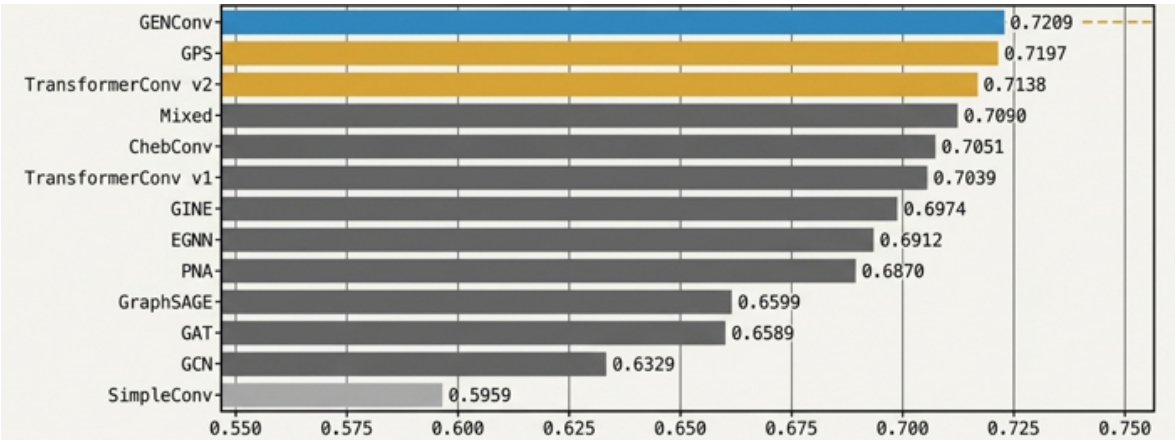


Figure 4.1: Comparison of the 13 evaluated models based on their Test Pearson correlation coefficient (R). Models are sorted in descending order of performance, from highest to lowest Test Pearson R.

4.5 Model Results

The thirteen evaluated GNN architectures demonstrated a wide range of performance on the PDBbind test set, revealing a clear ranking driven primarily by architectural power and edge feature support. At the top of the ranking, GENConv achieved the best overall performance with a test RMSE of 1.7689 kcal/mol and a test Pearson R of 0.7209, converging efficiently in only 45 epochs with the lowest training loss of all models (0.2376). GENConv achieved also the best validation RMSE of 1.7508 and the best validation Pearson R of 0.7326 among all evaluated models, a result we attribute to its differentiable balance between mean and max aggregation that adapts to the heterogeneous chemical environments of protein-ligand interfaces. GPS closely followed with the best test RMSE of 1.7641 kcal/mol and a Pearson R of 0.7197, demonstrating that combining local message passing with global self-attention effectively captures both short-range atomic contacts and long-range structural dependencies within the complex. Although GPS achieved the best test RMSE, GENConv achieved a higher test Pearson R (0.7209 vs 0.7197), which is the primary metric used for model selection in this work. GPS converged in 44 epochs with a training loss of 0.2974, demonstrating efficient learning dynamics comparable to GENConv. TransformerConv v2 outperformed TransformerConv v1 on ranking quality (R=0.7138 vs R=0.7039), demonstrating that the addition of batch normalization to the node embedding layer leads to measurable improvements in prediction consistency. The Mixed model (R=0.7090), ChebConv (R=0.7051), and TransformerConv both versions (0.7138 and 0.7039) surpassed the R=0.70 threshold, forming a strong upper tier of models that all share either native edge feature support, multi-hop receptive fields, or hybrid aggregation strategies. The middle tier comprises GINE (R=0.6974), PNA (R=0.6870), EGNN (R=0.6912), and GraphSAGE (R=0.6599), each achieving competitive but lower performance, GINE through its theoretically grounded sum aggregation,

PNA through its fixed multi-aggregator design whose lack of differentiable adaptation limited it relative to GENConv despite its theoretical richness, EGNN through its geometry-aware message computation that was partially redundant with the RBF distance encoding already present in the edge features, and GraphSAGE through its inductive sampling strategy that compensated only partially for the absence of edge features. The bottom tier is occupied by GAT (R=0.6589), GCN (R=0.6329), and SimpleConv (R=0.5959), all of which lack either edge feature support or learnable aggregation, with GCN requiring 151 epochs and GAT requiring 126 epochs to converge the two highest epoch counts in the study, and SimpleConv achieving the weakest results across all metrics, confirming that parameter-free mean aggregation is insufficient to capture the complexity of binding affinity from molecular graphs alone.

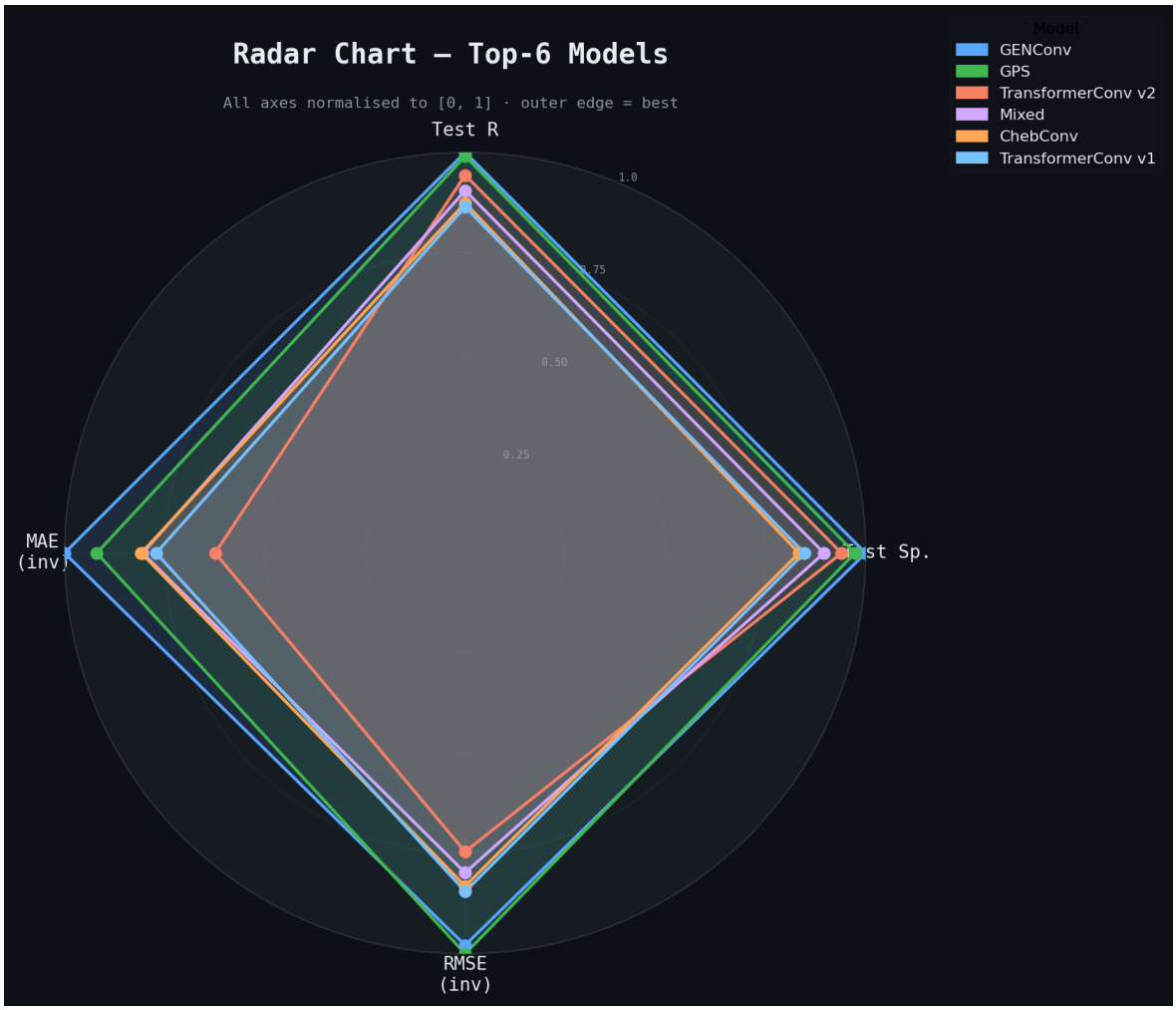


Figure 4.2: Radar chart comparing the six best-performing models across four evaluation metrics: RMSE, MAE, Pearson correlation coefficient (R), and Spearman rank correlation coefficient

4.6 Best Model Analysis

GENConv was selected as the best model based on its superior value of test Pearson R of 0.7209, which is the primary model selection criterion in this work. We present a detailed analysis of GENConv's training dynamics, prediction quality, and error distribution in the following subsections.

4.6.1 Training Curves

Figure 4.3 presents the training curves of the GENConv model across all training epochs. The training loss decreases steadily from an initial value of 1.3307 at epoch 1 to 0.2376 at the best checkpoint (epoch 45), demonstrating consistent and stable learning throughout training. The validation RMSE decreases from 2.2812 kcal/mol at epoch 1 to a best value of 1.7508 kcal/mol, while the validation Pearson R improves from 0.4624 to a peak of 0.7326. The learning rate is reduced progressively by the ReduceLROnPlateau scheduler from an initial value of 2×10^{-4} to 4.8×10^{-5} at the best checkpoint, reflecting the scheduler's adaptive response to validation plateau periods. Early stopping was triggered at epoch 67 after 15 consecutive epochs without improvement in validation Pearson R.



Figure 4.3: Training curves of the GENConv model showing training loss, validation RMSE, validation MAE, and validation Pearson R across all epochs.

4.6.2 Predicted vs Actual Scatter Plot

Figure 4.4 presents the scatter plot of predicted binding affinities against experimental ΔG values on the test set for the GENConv model. Each point represents one protein-ligand complex, with the experimental ΔG on the horizontal axis and the predicted ΔG on the vertical axis, both expressed in kcal/mol. The dashed diagonal line represents the ideal prediction where predicted values exactly match experimental values. The points cluster tightly around this diagonal, consistent with the high Pearson R of 0.7209 achieved on the test set. The predictions span a range from approximately -14 to -3 kcal/mol, closely matching the distribution of the experimental labels and demonstrating that the RangeAwareLoss penalty successfully prevents the prediction collapse observed in models trained with standard mean squared error.

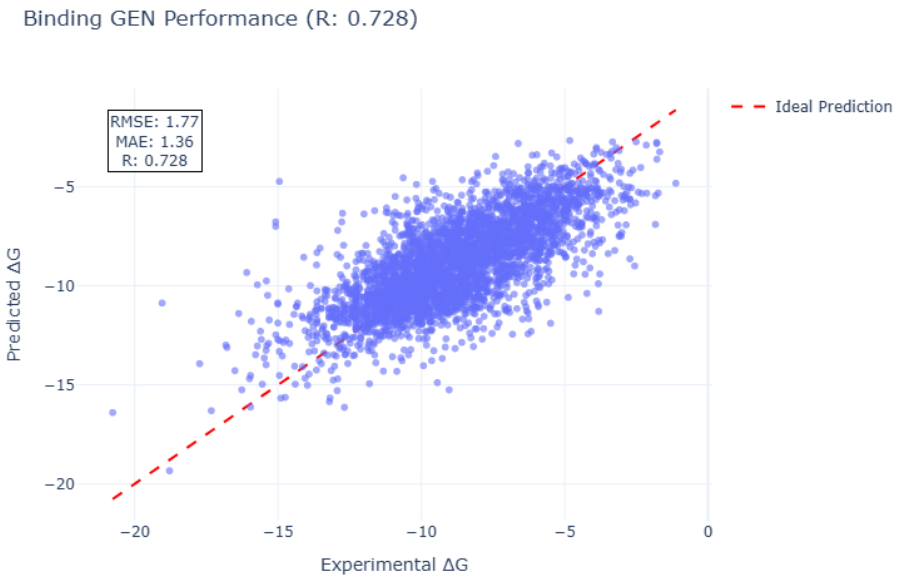


Figure 4.4: Scatter plot of GENConv predicted vs experimental ΔG (kcal/mol) on the test set. Each point represents one protein-ligand complex. The dashed line indicates perfect prediction.

4.6.3 Error Distribution

Figure 4.5 presents the distribution of prediction errors (predicted $\Delta G-$ experimental ΔG) for the GENConv model on the test set. The error distribution is approximately centered near zero, with a mean error close to 0 kcal/mol indicating the absence of systematic bias in either direction. The majority of prediction errors fall within the range of ± 2 kcal/mol, which is consistent with the test MAE of 1.3580 kcal/mol.

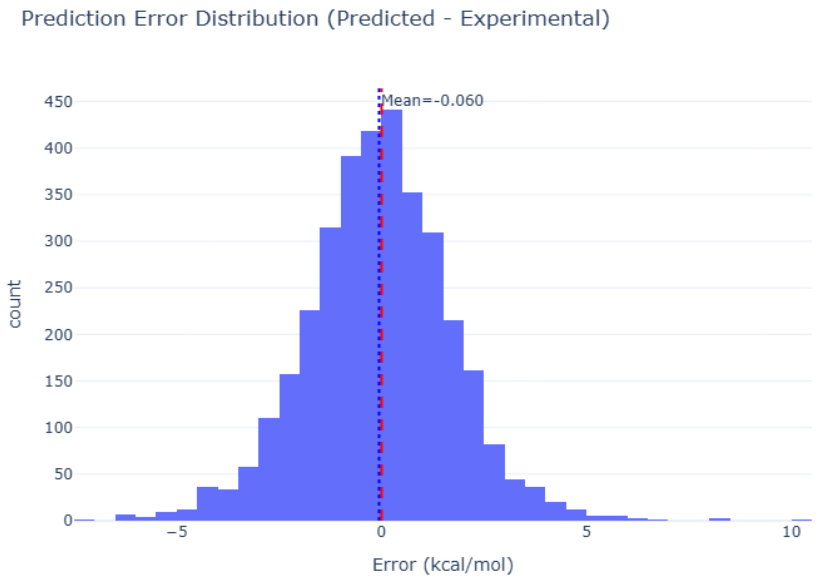


Figure 4.5: Distribution of prediction errors (predicted − experimental ΔG) for the GENConv model on the test set.

4.6.4 Comparative Prediction Quality Across Top Models

To evaluate whether the performance differences between the top three models reflect systematic behavioral differences or are simply artifacts of outlier sensitivity, Figure 4.6 presents side-by-side scatter plots of predicted versus experimental ΔG for GENConv, GPS, and TransformerConv v2.

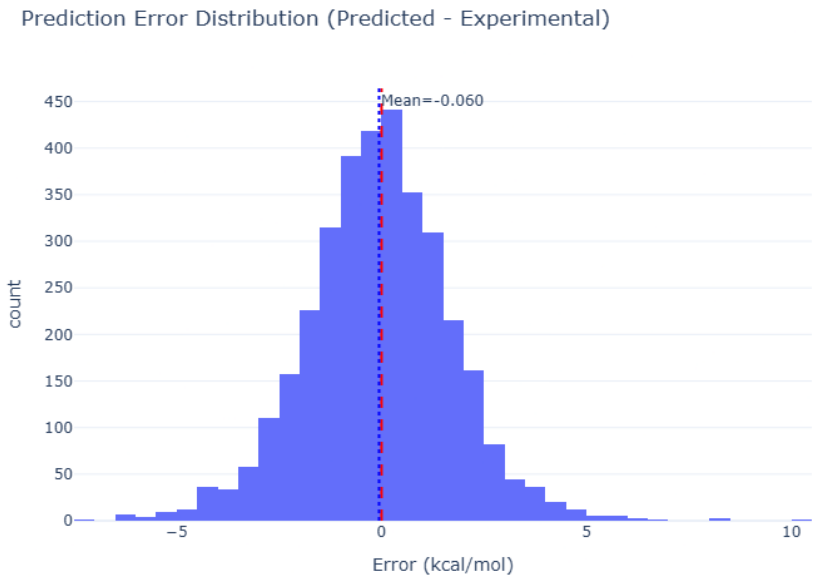


Figure 4.6: Distribution of prediction errors (predicted − experimental ΔG) for the GENConv model on the test set.

The three scatter plots reveal broadly similar prediction patterns across all top-performing models. All three models show tight clustering around the ideal diagonal in the central affinity range of approximately −12 to −5 kcal/mol, where training data is most abundant. Deviations from the diagonal are most pronounced at the extremes of the affinity distribution, particularly for very strong binders below −14 kcal/mol and very weak binders above −5 kcal/mol, consistent with the underrepresentation of these ranges in the PDBbind training set discussed in Section 4.8. The primary difference between the three models is visible in the spread of points around the diagonal. GENConv shows marginally tighter clustering, consistent with its higher Pearson R of 0.7209. GPS exhibits slightly higher variance for strong binders, which may explain why it achieves a better test RMSE (1.7641 kcal/mol) despite a marginally lower Pearson R, since RMSE is more sensitive to large individual errors. TransformerConv v2 shows a similar pattern to GPS but with slightly more scatter in the moderate affinity range. These observations suggest that the performance differences between the top models are small and consistent, and that no single model dominates across all regions of the affinity range.

4.7 Comparative Analysis with Related Works

Table 4.3 compares the performance of our best model (GENConv) against representative state-of-the-art methods from the binding affinity prediction literature on the PDBbind dataset.

Our best model achieves a test Pearson R of 0.7209, which is competitive with but below the state-of-the-art results reported in the literature. The performance gap is expected and can

Table 4.3: Comparison of our best model with related works on PDBbind.

Reference	Model	Dataset	RMSE (kcal/mol)	Pearson R
[17]	GraphDTA (GIN)	Davis	KIBA	1.35
0.78				
[17]	DGraphDTA (GCN)	Davis	KIBA	1.23
0.80				
[16]	DeepTGIN	PDBbind 2016	1.21	0.83
[15]	PLAPT	PDBbind	1.31	0.79
This work	GENConv	PDBbind	1.7689	0.7209

be attributed to several important differences in experimental setup. First, the related works listed in Table 4.3 use the PDBbind refined set, which is a smaller but higher-quality subset of PDBbind with stricter quality criteria, while our work uses the full general set after filtering, resulting in a larger and more diverse training distribution that is naturally more challenging. Second, methods such as DGraphDTA and DeepTGIN incorporate additional protein sequence encoders, typically CNN or Transformer-based that provide complementary sequence-level information beyond the binding pocket graph, while our approach encodes the protein exclusively through the 3D pocket graph. Third, methods such as PLAPT leverage large pre-trained language models (ProtBERT, ChemBERTa) that have been trained on massive molecular databases, providing rich prior knowledge that purely graph-based approaches trained from scratch cannot replicate.

Despite these differences, our result of Pearson R = 0.7209 is achieved using a unified GNN framework trained from scratch on raw molecular graphs without any pre-training, sequence encoders, or external molecular databases. This demonstrates that graph neural networks with appropriately designed molecular representations and loss functions are capable of learning meaningful binding affinity predictors from structural data alone, and that GENConv's generalized aggregation mechanism is particularly effective for this task.

4.8 Analysis of Factors Influencing Results

The experimental results reveal several important factors that consistently influence GNN performance on the binding affinity prediction task.

Edge feature support is a key differentiator. The clearest performance boundary in our results separates models that can incorporate edge features from those that cannot. Models without edge feature support GCN, GraphSAGE, ChebConv, and SimpleConv generally underperform models that natively process edge features. The only exception is ChebConv, which compensates for the absence of edge features through its K=3 multi-hop spectral filters.

This finding confirms that bond-level chemical information including bond type, conjugation, ring membership, and RBF-encoded interatomic distances carries essential information for predicting binding affinity that cannot be inferred from node features alone.

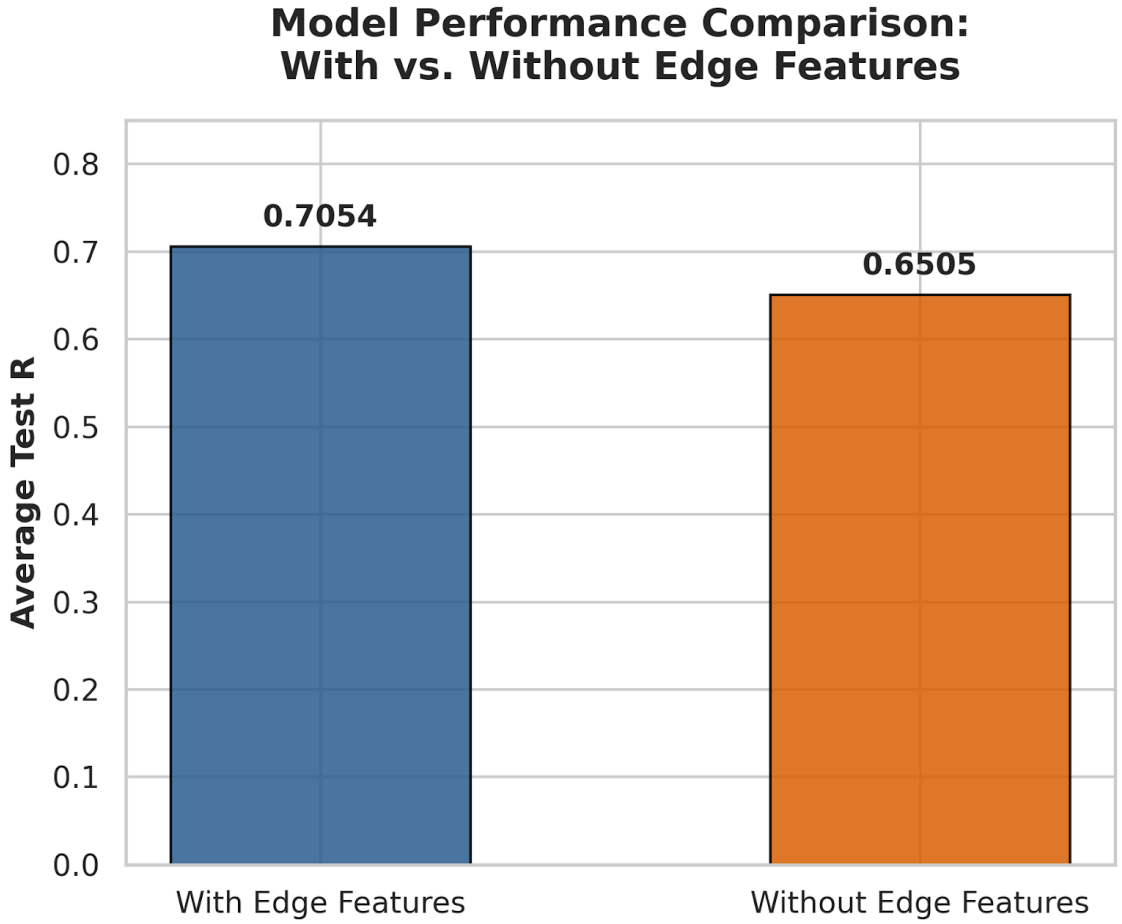


Figure 4.7: Comparison of the average Test Pearson correlation coefficient (R) for models with edge features and models without edge features.

Aggregation flexibility determines performance. The top-performing models GENConv and GPS both employ flexible, learned aggregation strategies rather than fixed operators. GENConv's flexible combination between mean and max aggregation allows it to adapt to the heterogeneous local chemical environments present in protein-ligand interfaces, where some binding interactions are dominated by a single key atom while others depend on collective neighborhood contributions. This flexibility appears to be the primary factor behind GENConv's superior performance compared to architectures with fixed aggregators such as GINE (sum) and PNA (fixed combination of four aggregators).

Global attention improves long-range modeling. GPS significantly outperforms pure message-passing models by combining local GINEConv layers with global multi-head self-attention. This result suggests that protein-ligand binding affinity is determined not only by local atomic

contacts but also by long range structural effects and electrostatic effects that extend across the entire complex.

Convergence speed is related to the expressive capacity of the architecture. The top-performing models (GENConv, GPS, GINE, PNA) consistently converge faster in fewer than 60 epochs and achieve lower training losses than the bottom performing models (GCN, GAT, GraphSAGE). This pattern suggests that more expressive architectures can find better optima more efficiently on the molecular graph representation used in this work, while less expressive architectures require more training iterations to partially makes up for their representational limitations.

4.9 Challenges and Obstacles

The development and evaluation of this work encountered several significant challenges that required careful solutions.

Computational resource constraints. All experiments were conducted on free colab and Kaggle GPU instances with session time limits and no persistent GPU memory between sessions on kaggle. Training thirteen GNN architectures on the full PDBbind dataset, each requiring up to 150 epochs, imposed a severe computational burden that required systematic checkpointing and session management. The implementation of a complete checkpoint system saving model state, optimizer state, scheduler state, and training history after every epoch was essential for resuming interrupted training runs without data loss.

Memory management for variable-size graphs. Protein-ligand complexes vary widely in size from small ligands with fewer than 20 atoms to large pocket graphs with some hundreds residue atoms resulting in highly variable graph sizes within each batch. This variability can cause out-of-memory errors on GPU when large complexes appear in the same batch as many smaller ones. We addressed this by enabling PyTorch's expandable memory segments through the `PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True` environment variable, which allows the CUDA allocator to reuse memory segments more efficiently across variable-size allocations.

Heterogeneous graph construction complexity. Constructing the unified heterogeneous graph that represents the ligand and protein pocket required careful handling of three edge types with different feature dimensions. Ensuring consistent 21-dimensional edge features across all edge types required careful implementation to avoid silent mismatches that could cause incorrect gradient flow.

Hyperparameter sensitivity of attention-based models. The TransformerConv and GPS models exhibited higher sensitivity to hyperparameter choices than simpler architectures, requiring careful tuning of dropout rates and learning rate schedules to achieve stable conver-

gence. In particular, TransformerConv v1 required significantly more epochs (89) than other comparable models, suggesting that the default hyperparameter configuration was not optimal for this architecture on the PDBbind dataset.

4.10 Conclusion

This chapter has presented a comprehensive experimental evaluation of thirteen graph neural network architectures on the protein-ligand binding affinity prediction task using the PDBbind dataset. All models were trained and evaluated under identical conditions within a unified base architecture, ensuring that the observed performance differences are attributable to the choice of graph convolution operator.

The main finding of this study is that GENConv, the generalized aggregation network, achieves the best overall performance with a test RMSE of 1.7689 kcal/mol and a test Pearson R of 0.7209. Its competitive convergence in only 45 epochs and the lowest training loss of all evaluated models suggest that its differentiable interpolation between mean and max aggregation is particularly well-suited to the heterogeneous molecular graph structure of protein-ligand complexes. GPS achieves the best test RMSE (1.7641 kcal/mol), confirming the value of combining local message passing with global attention. The bottom of the performance ranking is occupied by SimpleConv, GCN, and GAT, which are limited by their inability to leverage the full edge feature informations or adapt their aggregation strategies to the molecular graph structure. The analysis of factors influencing results reveals that edge feature support, aggregation flexibility, and global attention are the three most important architectural properties for this task, pointing to a clear direction for future improvement. Compared to state of the art methods, our results are competitive for a purely graph-based approach trained from scratch on raw structural data, and the performance gap with pre-training based methods such as PLAPT and DeepTGIN motivates the integration of molecular foundation models as a primary direction for future work.

GENERAL CONCLUSION

General CONCLUSION

The prediction of protein-ligand binding affinity is a fundamental problem in computational drug discovery, directly determining the ability of researchers to identify and prioritize promising medicinal compounds before committing to expensive experimental validation. Traditional approaches ranging from physics-based free energy calculations to empirical scoring functions have made significant contributions to this field but are limited either by their computational cost or by their dependence on manually engineered features that cannot capture the full complexity of atomic interactions at the binding interface. The development of graph neural networks has opened a new direction for structure-based affinity prediction, offering a framework that processes molecular structures in their natural graph representation and learns meaningful atomic interaction patterns directly from experimental data. This thesis sets out to systematically investigate which GNN architectures are best suited to this task, and to provide a comprehensive comparative benchmark that can guide future research in the field.

To achieve this objective, we designed and implemented a complete graph-based binding affinity prediction pipeline built on the PDBbind dataset. We constructed a unified heterogeneous molecular graph representation that encodes the ligand and the protein binding pocket as a single graph with three distinct edge types which are covalent ligand bonds, covalent pocket bonds, and non-covalent interaction edges ,where interaction edges are enriched with 16-dimensional RBF-encoded interatomic distances to provide a smooth and differentiable representation of the binding geometry. We implemented a shared base architecture comprising a linear node embedding, four GNN layers with residual connections, a dual pooling mechanism combining global mean pooling with gated attention pooling, and a three-layer MLP regression head. Within this unified framework, we evaluated thirteen distinct GNN convolution operators represented by GCN, GAT, GINE, GraphSAGE, GENConv, PNA, GPS, ChebConv, SimpleConv, TransformerConv v1, TransformerConv v2, EGNN, and a Mixed heterogeneous architecture, under identical training conditions using a custom RangeAwareLoss to prevent prediction collapse toward the mean of the training distribution.

The experimental results reveal a clear and interpretable performance hierarchy among the thirteen evaluated architectures. GENConv achieved the best overall performance with a test

Pearson R of 0.7209 and a test RMSE of 1.7689 kcal/mol, converging in only 45 epochs with the lowest training loss of all models. GPS achieved the best test RMSE of 1.7641 kcal/mol, confirming the value of combining local message passing with global self-attention for capturing long-range dependencies in protein-ligand complexes. TransformerConv v2 outperformed TransformerConv v1 on ranking quality (R=0.7138 vs R=0.7039), demonstrating that the addition of batch normalization to the node embedding layer leads to measurable improvements in prediction consistency. The Mixed Heterogeneous Architecture (R=0.7090) and ChebConv (R=0.7051) also achieved strong results, with ChebConv's K=3 multi-hop spectral filters proving remarkably effective despite the absence of edge feature support. At the lower end of the performance spectrum, GCN, GAT, and SimpleConv were limited by their inability to leverage the rich edge feature information or adapt their aggregation strategies to the heterogeneous molecular graph structure, with SimpleConv achieving the weakest results across all metrics (R=0.5959, RMSE=2.0250 kcal/mol). The analysis of per-range performance further revealed that all models struggle with very strong binders ($\Delta G < -14$ kcal/mol) and very weak binders ($\Delta G > -5$ kcal/mol) due to their underrepresentation in the training data, pointing to data imbalance as a primary challenge for future work.

The practical contribution of this work lies in demonstrating that a carefully designed graph representation of protein-ligand complexes with heterogeneous edge types, RBF-encoded geometric features, and a range-aware training objective enables purely graph-based GNN models trained from scratch to achieve competitive binding affinity prediction performance without relying on pre-trained language models, sequence encoders, or physics-based scoring components. The systematic comparison we provide across thirteen architectures offers the field a comprehensive and reproducible benchmark that quantifies the contribution of specific architectural properties edge feature support, aggregation flexibility, and global attention to binding affinity prediction performance on the PDBbind dataset. These findings provide practical guidance for researchers designing GNN-based drug discovery systems and establish GENConv's generalized aggregation mechanism as a particularly promising building block for molecular property prediction tasks.

Looking ahead, several directions show particular promise for extending this work. The integration of pre-trained molecular foundation models as initial node representations would likely produce the most significant performance improvements, as these models bring rich chemical knowledge learned from tens of millions of molecules that cannot be obtained from PDBbind alone. Incorporating full protein structure beyond the binding pocket, through hierarchical graph representations that capture both residue-level and atom-level information, would address a fundamental limitation of the current approach. Finally, ensemble methods combining the complementary strengths of GENConv and GPS could push performance further without requiring new architectural innovations.

Bibliography

[1] R. Wang, X. Fang, Y. Lu, and S. Wang, "The PDBbind database: Collection of binding affinities for protein-ligand complexes with known three-dimensional structures," *Journal of Medicinal Chemistry*, vol. 47, no. 12, pp. 2977–2980, 2004.

[2] M. K. Gilson and H.-X. Zhou, "Calculation of protein-ligand binding affinities," *Annual Review of Biophysics and Biomolecular Structure*, vol. 36, pp. 21–42, 2007.

[3] Q. U. Ain, A. Aleksandrova, F. D. Roessler, and P. C. Ballester, "Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening," *WIREs Computational Molecular Science*, vol. 5, no. 6, pp. 405–424, 2015.

[4] B. K. Shoichet, "Virtual screening of chemical libraries," *Nature*, vol. 432, no. 7019, pp. 862–865, 2004.

[5] M. Jiang, Z. Li, S. Zhang, S. Wang, X. Wang, Q. Yuan, and Z. Wei, "Drug-target affinity prediction using graph neural network and contact maps," *RSC Advances*, vol. 10, no. 35, pp. 20701–20712, 2020.

[6] T. Nguyen, H. Le, T. P. Quinn, T. Nguyen, T. D. Le, and S. Venkatesh, "GraphDTA: Predicting drug–target binding affinity with graph neural networks," *Bioinformatics*, vol. 37, no. 8, pp. 1140–1147, 2021.

[7] R. Meli, G. Morris, and P. C. Biggin, "Scoring functions for protein-ligand binding affinity prediction using structure-based deep learning: A systematic review," *Frontiers in Bioinformatics*, vol. 2, 2022.

[8] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions," *Journal of Molecular Biology*, vol. 161, no. 2, pp. 269–288, 1982.

[9] P. J. Goodford, "A computational procedure for determining energetically favorable binding sites on biologically important macromolecules," *Journal of Medicinal Chemistry*, vol. 28, no. 7, pp. 849–857, 1985.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] M. Hajij et al., "Topological deep learning: Going beyond graph data," *arXiv preprint arXiv:2206.00606*, 2023.

[12] M. Chatzianastasis, J. Lutzeyer, G. Dasoulas, and M. Vazirgiannis, "Graph ordering attention networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 7006–7014, 2023.

[13] X. Liu, J. Chen, and Q. Wen, "A survey on graph classification and link prediction based on GNN," *arXiv preprint arXiv:2307.00865*, 2023.

[14] A. Ghali, A. Anter, and A. Bhatt, "A comprehensive survey on graph convolutional networks for graph-based machine learning," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[15] T. Rose, N. Monti, N. Anand, and T. Shen, "PLAPT: Protein-ligand binding affinity prediction using pre-trained transformers," *bioRxiv*, 2024.

[16] G. Wang, H. Zhang, M. Shao, Y. Feng, C. Cao, and X. Hu, "DeepTGIN: A novel hybrid multimodal approach using transformers and graph isomorphism networks for protein-ligand binding affinity prediction," *Journal of Cheminformatics*, vol. 16, no. 1, 2024.

[17] G. A. Abdelkader and J.-D. Kim, "Advances in protein-ligand binding affinity prediction via deep learning: A comprehensive study of datasets, data preprocessing techniques, and model architectures," *Current Drug Targets*, vol. 25, pp. 1041–1065, 2024.

[18] W. Jia, D. Zhao, T. Shen, C. Su, C. Hu, and Y. Zhao, "A new optimized GA-RBF neural network algorithm," *Computational Intelligence and Neuroscience*, vol. 2014, 2014.

[19] V. G. Satorras, E. Hoogeboom, and M. Welling, "E(n) equivariant graph neural networks," in *Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, pp. 9323–9332, 2021.

[20] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1548–1554, 2021.

[21] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "DeeperGCN: All you need to train deeper GCNs," *arXiv preprint arXiv:2006.07739*, 2020.

[22] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 13260–13271, 2020.

[23] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph Transformer," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[25] F. Wu, T. Zhang, A. H. de Souza Jr., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97, pp. 6861–6871, 2019.

[26] G. van Rossum and F. L. Drake, *Python Reference Manual*, PythonLabs, 2002.

[27] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[28] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[29] A. P. Bento et al., "An open source chemical structure curation pipeline using RDKit," *Journal of Cheminformatics*, vol. 12, no. 51, 2020.

[30] E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Apress, Berkeley, CA, pp. 59–64, 2019.

[31] K. Bönisch and L. Losaria, "Kaggle Chronicles: 15 years of competitions, community and data science innovation," *arXiv preprint arXiv:2511.06304*, 2025.

[32] C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

[33] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] S. Wan, X. Zhang, X. Xue, and P. V. Coveney, "On the reliability of AI methods in drug discovery: Evaluation of Boltz-2 for structure and binding affinity prediction," *arXiv preprint arXiv:2603.05532*, 2026. [Online]. Available: https://arxiv.org/abs/2603.05532

Webography

[URL1] PDBbind Database.
http://www.pdbbind.org.cn
Accessed: June 2026.

[URL2] PyTorch Official Documentation.
https://pytorch.org/docs
Accessed: June 2026.

[URL3] PyTorch Geometric Documentation.
https://pytorch-geometric.readthedocs.io
Accessed: June 2026.

[URL4] RDKit Open-Source Cheminformatics.
https://www.rdkit.org
Accessed: June 2026.

[URL5] Kaggle Platform.
https://www.kaggle.com
Accessed: June 2026.

[URL6] Google Colaboratory.
https://colab.research.google.com
Accessed: June 2026.

[URL7] Protein Data Bank (PDB).
https://www.rcsb.org
Accessed: June 2026.