الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

University Yahia Farès Medea

Faculty of Sciences

قسم الرياضيات والإعلام الآلي

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

Speciality: Applied Artificial Intelligence

Thesis for obtaining the Master's degree

THEME

DeepToxiGraph: An Integrated Graph Neural Network Framework for Toxicity Prediction

Prepared and Presented by

ALLICHE Mohamed Amine.

MIHOUBI Mohamed Abdelhak.

Proposed and Directed

Dr. HENTABLI Hamza.

Dr. MAZARI Reda.

2025-2026

ACKNOWLEDGMENT

First and foremost, I would like to express my gratitude to ALLAH for allowing me to accomplish this project.

We would like to thank my supervisors, Dr. Hentabli Hamza and Dr. Mazari Reda, for their guidance, support, feedback, and encouragement throughout this study. Their expertise and advice played an important role in the completion of this study.

We would also like to thank everyone who contributed to the completion of this study, directly or indirectly.

DEDICATION

I dedicate this modest work

To my dear mother and father.

To my aunts.

To my brother Zinou.

To my sisters Hafsa.

To all my family members without exception.

To my professors and supervisors.

To the little ones in my life.

To all my friends.

To everyone who knows me.

I would like to thank all of those who supported me and were with me in all this time.

M. Abdelhak

DEDICATION

In the name of Allah, the Most Gracious, the Most Merciful.

I dedicate this work to my father and my mother for their unconditional love, endless sacrifices, patience, and support throughout every stage of my life. Without them, this achievement would not have been possible.

To my dear sisters — Ikram, Fatima, Yosra, Rihab, Asma, and Yasmin — thank you for your encouragement, support, and for always standing by my side.

To my grandmother and my grandfather, for their prayers, kindness, and wisdom.

To my uncles and aunts, especially Bachir and Wahiba, thank you for your trust, motivation, and continuous encouragement.

Special thanks to my partner and my best friend, with whom I shared this journey, the hard work, the challenges, and the unforgettable moments throughout this year.

I also express my sincere gratitude to everyone who supported me, believed in me, and encouraged me during this journey.

Finally, I dedicate this work to all those who trusted my abilities and inspired me to always move forward and never give up.

Thank you all.

A. Mohamed Amine

ABSTRACT

The problem of predicting the toxicity of chemical compounds is among the most important problems in the field of drug discovery. Most conventional approaches to modeling the relationship between the chemical structures of compounds and their toxicities encode the structures of those chemicals as vectors of features, thereby discarding information about the relationship between the chemicals and their structures during the encoding process. As a result, the models trained using such encodings tend to have limited predictive power when applied to diverse libraries of chemical compounds.

In this work, DeepToxiGraph is introduced as a deep learning framework for the task of toxicity prediction that uses several different graph neural network architectures to encode chemical compounds and predict their toxicity. Through a series of ablation studies, each of which eliminates a certain feature of the models, the relative importance of various features of graph neural networks is studied. Based on these studies, a novel hybrid model that combines graph, fingerprint, and descriptor representations is proposed. The hybrid model achieves an R^2 of 0.6307 and an RMSE of 0.4161 on the mouse intraperitoneal LD_{50} test set. This entire framework is compiled into a web site that allows chemistry scientists to predict the toxicity of chemical compounds by simply inputting their SMILES notations, without the need to learn any programming language or skills.

Key words

Molecular toxicity prediction, QSTR, Graph Neural Network, GNN, GraphSAGE, Graph Transformer, Deep learning, Hybrid model, LD_{50}.

RÉSUMÉ

La prédiction de la toxicité des composés chimiques figure parmi les problèmes les plus importants en pharmacologie. La plupart des approches conventionnelles de modélisation de la relation entre la structure chimique des composés et leur toxicité encodent ces structures sous forme de vecteurs de caractéristiques, perdant ainsi des informations sur la relation entre les composés et leurs structures lors de l'encodage. De ce fait, les modèles entraînés avec de tels encodages présentent généralement un pouvoir prédictif limité lorsqu'ils sont appliqués à diverses bibliothèques de composés chimiques.

Dans ce travail, DeepToxiGraph est introduit comme un cadre d'apprentissage profond pour la prédiction de la toxicité. Il utilise différentes architectures de réseaux neuronaux graphiques pour encoder les composés chimiques et prédire leur toxicité. Grâce à une série d'études d'ablation, éliminant chacune une caractéristique spécifique des modèles, l'importance relative des différentes caractéristiques des réseaux neuronaux graphiques est étudiée. Sur la base de ces études, un nouveau modèle hybride combinant représentations par graphe, empreinte digitale et descripteur est proposé. Le modèle hybride atteint un coefficient de détermination (R^2) de 0,6307 et une erreur quadratique moyenne (RMSE) de 0,4161 sur l'ensemble de données de test de DL_{50} par voie intrapéritonéale chez la souris. L'ensemble de ce système est intégré à un site web permettant aux chimistes de prédire la toxicité de composés chimiques en saisissant simplement leurs notations SMILES, sans aucune connaissance en programmation.

Mots Clés

Prédiction de toxicité moléculaire, QSTR, Réseau de neurones graphiques, GNN, GraphSAGE, Graph Transformer, Apprentissage profond, Modèle hybride, LD_{50}.

ملخص

تعدّ مشكلة التنبؤ بسمية المركبات الكيميائية من أهم المشكلات في مجال اكتشاف الأدوية. تعتمد معظم الطرق التقليدية لنمذجة العلاقة بين البنية الكيميائية للمركبات وسميتها على ترميز هذه البنى الكيميائية كمتجهات من الخصائص، متجاهلةً بذلك معلوماتٍ مهمة حول العلاقة بين المركبات الكيميائية وبنيتها أثناء عملية الترميز. ونتيجةً لذلك، تميل النماذج المُدرَّبة باستخدام هذه الترميزات إلى امتلاك قدرة تنبؤية محدودة عند تطبيقها على مكتبات متنوعة من المركبات الكيميائية.

في هذا العمل، نُقدّم DeepToxiGraph كإطار عمل للتعلم العميق مُخصَّص لمهمة التنبؤ بالسمية، حيث يستخدم عدة بنى مختلفة لشبكات عصبية بيانية لترميز المركبات الكيميائية والتنبؤ بسميتها. ومن خلال سلسلة من دراسات الاستبعاد، التي تُزيل كلٌّ منها خاصيةً مُحدَّدة من النماذج، تمّت دراسة الأهمية النسبية لمختلف خصائص الشبكات العصبية البيانية. واستنادًا إلى هذه الدراسات، نقترح نموذجًا هجينًا جديدًا يجمع بين تمثيلات الرسم البياني، وبصمات الأصابع، والوصف. يحقق النموذج الهجين معامل تحديد (R^2) قدره 0.6307 وجذر متوسط مربع الخطأ (RMSE) قدره 0.4161 على مجموعة اختبار الجرعة المميتة LD_{50} داخل الصفاق للفئران. وقد جُمع هذا الإطار بالكامل في موقع إلكتروني يُمكّن علماء الكيمياء من التنبؤ بسمية المركبات الكيميائية بمجرد إدخال رموز SMILES الخاصة بها، دون الحاجة إلى تعلم أي لغة برمجة أو مهارات برمجية.

الكلمات الدالة

التنبؤ بالسُّمِّية الجزيئية، العلاقة الكمية بين البنية والسُّمِّية (QSTR)، شبكات العصبونات البيانية (GNN)، خوارزمية GraphSAGE، المحوّل البياني (Graph Transformer)، التعلّم العميق، النموذج الهجين، الجرعة المميتة الوسطية (LD_{50}).

Table of Contents

ABSTRACT I

Key words I

RÉSUMÉ II

ملخص III

الكلمات الدالة III

List of Tables VIII

List of Figures IX

List of Acronyms XI

Glossary XII

General Introduction 2

CHAPTER 1 AN OVERVIEW OF THE TOPIC 4

1.1 Introduction 4

1.2 Overview 5

1.3 Background of the Problem 5

1.4 Problem Statement 6

1.5 Research Objectives 7

1.6 Scope of the Research 7

1.7 Thesis Outline 8

1.8 Summary 9

CHAPTER 2 LITERATURE REVIEW 10

2.1 Introduction 11

2.2 Drug Design 11

2.2.1 Computer-Aided Drug Design 11

2.2.1.1 Structure-Based Computer-Aided Drug Design (SB-CADD) 12

2.2.1.2 Ligand-Based Computer-Aided Drug Design (LB-CADD) 12

2.3 Molecular Representation 12

2.3.1 SMILES 12

2.3.2 Molecular Fingerprints 13
2.3.3 Physicochemical Descriptors 13
2.3.4 Molecular Graphs 14
2.4 Graph-Theoretic Representations 15
2.4.1 Adjacency Matrix 16
2.4.2 Degree Matrix 17
2.4.3 Laplacian Matrix 17
2.4.4 Symmetric Normalised Laplacian 18
2.4.5 Incidence Matrix 18
2.5 Toxicity Prediction 18
2.6 Prior Research on Toxicity Prediction 19
2.6.1 Classical QSAR Models 19
2.6.2 Deep Neural Networks 19
2.7 Machine Learning 20
2.8 Deep Learning 21
2.8.1 Convolutional Neural Networks 22
2.8.2 Graph Neural Networks 24
2.8.2.1 Graph Convolutional Network (GCN) 27
2.8.2.2 GraphSAGE 28
2.8.2.3 Graph Attention Network (GAT) 29
2.8.2.4 Graph Isomorphism Network (GIN) 29
2.8.2.5 Graph Transformer 29
2.9 Transformers and Attention Mechanisms 30
2.9.1 Scaled Dot-Product Attention 32
2.9.2 Multi-Head Attention 33
2.10 Batch Normalization 34
2.11 Dropout 35
2.12 Readout Functions 35
2.13 Summary 35

CHAPTER 3 METHODOLOGY 37
3.1 Introduction 38
3.2 Research Framework 38
3.3 Dataset and Data Splitting 39
3.3.1 The LD_{50} Dataset 39
3.3.2 Molecular Graph Construction 41
3.3.3 Train and Test Split 41
3.4 Model Development 41
3.4.1 Stage 1 Finding the Best Readout Function 41
3.4.2 Stage 2 Comparing GNN Architectures 43
3.4.3 Stage 3 Depth Search, Batch Normalization Placement and Scale-Up 44
3.4.3.1 Depth Search 44
3.4.3.2 Batch Normalization Placement Study 45
3.4.3.3 Final GNN Configuration 47
3.4.4 Stage 4 Fingerprint Based 2D CNN Models 49
3.4.4.1 Fingerprint Representation 49
3.4.4.2 Individual Fingerprint Evaluation 50
3.4.4.3 Fingerprint Fusion: Concatenated 3-Channel 2D CNN 50
3.4.5 Stage 5 Physicochemical Descriptor MLP Baseline 51
3.4.5.1 Descriptor Computation 51
3.4.6 Stage 6 DeepToxiGraph Hybrid Model 52
3.4.6.1 Architecture Overview 53
3.4.6.2 GNN Branch 53
3.4.6.3 Fingerprint MLP Branch 54
3.4.6.4 Descriptor MLP Branch 54
3.4.6.5 Concatenation Fusion and Prediction Head 54
3.4.6.6 Training Configuration 55
3.5 Evaluation Metrics 55

3.6 Summary 56
CHAPTER 4 RESULTS AND DISCUSSIONS 58
4.1 Introduction 59
4.2 Stage 1 Readout Strategy Results 59
4.3 Stage 2 GNN Architecture Results 60
4.4 Stage 3 62
4.4.1 Depth Results 62
4.4.2 Batch Normalization Placement Results 63
4.4.3 Final GNN Configuration Results 64
4.5 Stage 4 Fingerprint-Based 2D CNN Models 65
4.5.1 Individual Fingerprint Results 65
4.5.2 Fingerprint Fusion Results 66
4.6 Stage 5 Physicochemical Descriptor MLP 66
4.7 Stage 6 DeepToxiGraph Hybrid Model Results 67
4.8 Comparison with Related Work 69
4.9 Application 72
4.10 Summary 75
General conclusion 78
REFERENCES 80

List of Tables

Table 1: Description of the LD_{50} dataset. 39
Table 2: Sample rows from the LD_{50} dataset sourced from the TOXRIC database. 40
Table 3: The nine readout strategies evaluated in Stage 1. 42
Table 4: Batch normalization placement configurations for Graph Transformer-4L. 46
Table 5: Test R^2 scores for all readout strategies. 60
Table 6: Architecture comparison results. 61
Table 7: Depth ablation results sorted by test R^2. 62
Table 8: Batch normalization placement results for GT-4L, sorted by test R^2. 63
Table 9: Performance of nine fingerprint types on the LD_{50} test set using the 2D CNN. Sorted by R^2 descending. 65
Table 10: Comparison of individual fingerprint models and the concatenated 3-channel CNN fusion on the LD_{50} test set. 66
Table 11: Final test-set performance of the Descriptor MLP. 67
Table 12: Final performance comparison standalone GNN models vs Hybrid. 69
Table 13: Comparison of DeepToxiGraph with related LD_{50} prediction models. 70

List of Figures

Figure 1: SMILES Representation. 13
Figure 2: Molecular descriptor representations RDKit [1]. 14
Figure 3: Molecular Graph. 15
Figure 4: Directed and undirected graphs [2]. 16
Figure 5: Adjacency matrix [3]. 17
Figure 6: Deep learning is a subfield of machine learning, which is in turn a subfield of artificial intelligence [4]. 21
Figure 7: Typical CNN architecture [5]. 22
Figure 8: 3×3 convolutional filter [5]. 23
Figure 9: Atom feature encoding [6]. 25
Figure 10: CNN versus GNN [7]. 26
Figure 11: CNN and GNN updates [7]. 27
Figure 12: Stacked GCN layers [7]. 28
Figure 13: GNN versus Transformer [8]. 30
Figure 14: Attention mechanism motivation [9]. 31
Figure 15: Scaled dot-product attention [9]. 33
Figure 16: Multi-head self-attention [9]. 34
Figure 17: Overview of the six-stage DeepToxiGraph methodology. 39
Figure 18: Nine readout strategies. 43
Figure 19: Stage 2: four GNN architectures. 44
Figure 20: Stage 3a: depth search for Graph Isomorphism Network and Graph Transformer at 4, 5, and 6 layers. 45
Figure 21: Ten batch normalization placement configurations evaluated on Graph Transformer-4L. 47
Figure 22: GraphTransformer_4L architecture. 48
Figure 23: CNN architecture used for all fingerprints. 49
Figure 24: Fusion 3-channel model combining Avalon, RDKit5, and ECFC4 fingerprints. 51
Figure 25: Descriptor MLP architecture. 52
Figure 26: DeepToxiGraph hybrid model. 53
Figure 27: Test R^2 comparison of nine readout strategies on a 3-layer GCN. 59
Figure 28: Test R^2 for GNN architectures with Max + Add readout. 61

Figure 29: Depth ablation study comparing GIN and GraphTransformer across 4, 5, and 6 message-passing layers using Max + Add readout (no BN). 62
Figure 30: Predicted vs. True LD_{50} scatter plot for GraphTransformer. 64
Figure 31: Predicted vs. true log LD_{50} scatter. 67
Figure 32: Scatter plot for Hybrid_GT4L (R^2 = 0.6307, RMSE = 0.4161, r = 0.7952). 68
Figure 33: Hybrid Model: Ranked predictions sorted by ascending absolute error, showing the top 29 most accurate molecule predictions. 71
Figure 34: DeepToxiGraph web application interface. 72
Figure 35: 3D molecular structure of the input molecule. 73
Figure 36: 2D molecular structure of the input molecule. 74
Figure 37: Prediction output of the DeepToxiGraph web site. 74

List of Acronyms

1D — One Dimensional

2D — Two Dimensional

3D — Three Dimensional

BN — Batch Normalization

CADD — Computer-Aided Drug Design

CNN — Convolutional Neural Network

DL — Deep Learning

ECFP — Extended-Connectivity Fingerprint

FC — Fully Connected

GAT — Graph Attention Network

GCN — Graph Convolutional Network

GIN — Graph Isomorphism Network

GNN — Graph Neural Network

GT — Graph Transformer

LB-CADD — Ligand-Based Computer-Aided Drug Design

LD_{50} — Lethal Dose 50% — the dose lethal to 50% of a test population

MACCS — Molecular ACCess System fingerprints

ML — Machine Learning

MLP — Multilayer Perceptron

QSAR — Quantitative Structure–Activity Relationship

QSTR — Quantitative Structure–Toxicity Relationship

RDKit — Open-Source Cheminformatics Library

RMSE — Root Mean Squared Error

R^2 — Coefficient of Determination

Pearson's r — Pearson Correlation Coefficient

SB-CADD — Structure-Based Computer-Aided Drug Design

SMILES — Simplified Molecular Input Line Entry System

Glossary

Atom

The smallest unit of a chemical element that retains the properties of that element composed of a nucleus surrounded by electrons.

Chemical Database

A structured repository specifically designed to store chemical compound data, including molecular structures, properties, and biological activities.

Graph

A mathematical structure G = (V, E) composed of a set of vertices V and a set of edges E, used to represent entities and their pairwise relationships.

LD_{50}

The lethal dose at which 50% of a test population is expected to die; one of the most widely used acute toxicity endpoints in pharmaceutical research.

Message Passing

The computational process in GNNs by which each node aggregates information from its neighboring nodes across successive layers of the network.

Molecule

The smallest structural unit of a substance that retains the chemical properties of that substance composed of one or more atoms linked by chemical bonds.

Molecular Descriptor

A numerical value derived from the chemical structure of a molecule, used to characterize its properties for computational modeling.

Molecular Fingerprint

A binary or numerical vector encoding the presence or absence of specific structural substructures within a molecule.

Node Embedding

A learned vector representation of a node in a graph that encodes both its own features and the structural context of its neighborhood.

Readout Function

A pooling operation that aggregates the individual node embeddings of a molecular graph into a single fixed-size vector representing the whole molecule.

SMILES

A line notation for encoding the structure of a chemical molecule as a string of ASCII characters, used as a standard exchange format in chemical databases.

Similarity Coefficient

A measure used to quantify the degree of structural resemblance between two molecular structures, commonly the Tanimoto coefficient.

QSTR Model

A mathematical model that correlates the structural features of a molecule with its measured toxicity, used to predict the toxicity of novel compounds.

Coefficient of Determination (R^2)

A statistical measure that expresses the proportion of the variance in the observed values that is explained by a model's predictions. It ranges from $-\infty$ to 1, where 1 indicates perfect prediction and 0 indicates a model that performs no better than predicting the mean of the data. In this work, R^2 is the primary criterion used to select the final model.

Root Mean Squared Error (RMSE)

The square root of the average of the squared differences between the predicted and the true values. Expressed in the same units as the target variable, it is directly interpretable and penalizes large prediction errors more heavily than small ones; lower values indicate better predictive performance.

Pearson's r

The Pearson correlation coefficient, a measure of the linear correlation between the predicted and the true values. It ranges from -1 to 1, where 1 denotes a perfect positive linear relationship, 0 denotes no linear relationship, and -1 denotes a perfect negative relationship. It indicates how well a model ranks compounds from least to most toxic.

GENERAL INTRODUCTION

General Introduction

Every year, the pharmaceutical industry synthesizes hundreds of thousands of new chemical compounds in the quest to produce new medicines that are both safer and more effective than existing drugs. Behind each of these synthesized chemicals is a question that cannot be avoided: is it toxic, and to what degree? The answer to this question requires the performance of toxicological studies on laboratory animals, raising both ethical and financial concerns. As the number of possible chemical compounds that can be synthesized reaches into the billions of chemicals each year, a growing gap exists between the development of those chemicals and the capacity of the pharmaceutical industry and its regulators to test for toxicity.

One way of overcoming this challenge is through the use of computational models to predict the potential toxicity of each chemical compound that is synthesized. Such models are referred to as Quantitative Structure-Toxicity Relationship (QSTR) models, and have been developed and utilized for decades within the pharmaceutical and chemical regulatory industries around the world. Often, these models use the language of graph theory to represent the chemical compounds that are to be evaluated for potential toxicity. Such representations utilize the chemical atoms within the compound as the nodes of a graph, and the chemical bonds between those atoms as the edges of the graph. Graph Neural Networks (GNNs) are deep learning models that are able to understand the structures of those graphs, allowing their use within toxicity prediction models.

It is this convergence of fields within the pharmaceutical, information, and chemical sciences that motivates the development of DeepToxiGraph. Most existing models for predicting the toxicity of chemical compounds have relied upon the use of a fixed representation of the chemical compound. GNNs eliminate the need for such a fixed pre-computed representation, since they learn the molecular representation directly from the graph structure, leading to the question of whether such models can outperform existing methods on a large and chemically-diverse dataset of toxicity measurements.

In this thesis, we develop DeepToxiGraph, a deep learning framework designed to answer that question. Our work systematically evaluates several GNN architectures for their ability to accurately predict the LD_{50} [10], [11] of chemical compounds introduced into the peritoneal cavity of mice, identifies which architecture performs best on this task, and then proposes a novel hybrid model that combines the strengths of graph, fingerprint,

and descriptor representations into a single predictive framework. The entire framework is deployed as a user-friendly web site that we built to make toxicity prediction accessible to researchers without any programming experience.

This thesis is comprised of four main chapters. Chapter One presents an overview of the problem that was evaluated, the motivation behind the development of DeepToxiGraph, and the objectives to be met through this framework. Chapter Two presents the theoretical foundations of DeepToxiGraph, including discussions of the representations of chemicals, graph theory, deep learning, and GNNs. Chapter Three presents the experimental methodology that was employed to develop and evaluate the model. Chapter Four presents an overview of the DeepToxiGraph implementation, the results of each of the experiments that were performed with DeepToxiGraph, and a discussion of those results.

CHAPTER 1

AN OVERVIEW OF THE TOPIC

1 Introduction

This chapter introduces the topic addressed in this thesis and establishes the context, motivation, and objectives of the DeepToxiGraph framework. New chemical compounds are discovered every year, but their potential toxicities remain unresolved. Testing chemicals in the laboratory is no longer a sustainable method of determining their toxicities, given the vast chemical space that can be synthesized and the cost and ethical concerns of laboratory testing. With the growth of information technology, computational methods for predicting the toxicity of chemicals have become of increasing interest to researchers. One such method is Quantitative Structure-Toxicity Relationship (QSTR) modeling [10], [12], which uses the chemical structure of a compound to predict its toxicity, avoiding the need to test the chemical in the laboratory. As a result, computational methods such as QSTR modeling have become an established part of the chemical research field.

Despite their popularity, most QSTR models have remained the same for decades. Chemical structures are represented as lengthy vectors of chemical descriptors or fingerprints [13], [14], after which various regression or machine learning models are trained to learn the relationship between the chemical structure and toxicity. The limitation of this representation and modeling approach is that the relational information regarding the atoms within a chemical compound is lost during the creation of the chemical representation. The model cannot understand the relationship between the atoms of a compound and its toxicity.

Graph Neural Networks (GNNs) offer a way of overcoming this limitation of QSTR modeling. Chemical compounds can be represented as graphs of molecules, with the atoms within the molecule represented as nodes and the bonds between those atoms represented as edges. GNNs use message passing between these nodes to learn about the chemical structure of the molecule and create a representation of that molecule [6]. These representations have led to advances in the prediction of various chemical properties, but the use of GNNs for the prediction of chemical toxicity is underexplored.

DeepToxiGraph is a deep learning framework that aims to fill this gap in the chemical research field. It compares five different GNN architectures for the task of LD_{50} prediction [10], [11], determines which one performs best, and then creates a new, hybrid architecture that combines the best GNN architecture with chemical fingerprints and

descriptors. This new model is then deployed as a web application that allows any chemical researcher to determine the toxicity of a chemical compound from its SMILES string without writing a single line of code.

The remainder of this chapter is organized as follows. Section 1.2 gives an overview of the DeepToxiGraph framework and its experimental stages. Section 1.3 presents the background of the problem and the evolution of computational toxicity prediction. Section 1.4 states the research problem and the gaps in the existing literature that motivate this work. Section 1.5 defines the objectives of the study, and Section 1.6 specifies its scope and limitations. Section 1.7 outlines the structure of the thesis, and Section 1.8 summarizes the chapter.

2 Overview

DeepToxiGraph consists of six experimental stages that are connected to each other, where the completion of one stage enables the start of the next. Each of the stages addresses a specific design question. The first stage determines how the atomic embeddings should be read out into a single molecular embedding. Once this part of the system is established, the second stage compares four different GNN architectures on the same dataset.

The two best-performing architectures from the second stage are analyzed in the third and fourth stages to find the best architecture for which to apply batch normalization to create a regularized standalone GNN. In the fifth stage, a separate branch of the DeepToxiGraph model is trained using nine different fingerprint types represented as 2D images. Finally, a third branch of the model is trained using 208 physicochemical descriptors for the standalone MLP baseline, and 208 for the hybrid model. The three branches are combined in the final stage to create a hybrid DeepToxiGraph model.

3 Background of the Problem

The connection between the molecular structure and biological effect has been studied systematically since the 1960s. The idea that molecules with similar structures tend to have similar biological effects is the founding principle of a field that today encompasses cheminformatics, computational chemistry, and machine learning approaches to understanding the effects of molecular structure [13], [15].

The earliest attempts to create quantitative structure-activity relationship (QSTR) models [16] utilized linear methods to analyze small sets of molecules. Non-linear methods were later introduced as means of improving the performance of the models, but still required the consideration of various pre-computed features of the molecules. Regardless of the machine learning models that are applied to a dataset of molecules, the model will be no more informative than the features of the molecules that are considered in its construction [10], [17].

Deep learning approaches to modeling biological effects utilized various neural network models trained on sets of pre-computed structural features of the molecules (often based on extended-connectivity fingerprints [14]) [18]. The use of machine learning approaches like multi-task learning improved some of the deep learning approaches. However, in each of these deep learning models, the structure of the molecule was still fed into the model as a pre-computed vector of features, and the information encoded into that vector was dependent upon the chosen fingerprinting algorithm alone.

Deep learning models based on graphs (GNNs) allow the molecule to be fed into the model as a graph, and for the model to learn its own representation of that molecule and its features. The message passing algorithm between the nodes of the molecule graph allows the model to effectively learn what features of the graph are associated with the various biological effect labels associated with each molecule [6]. Thus, the features learned by the model from the example molecules and their biological effect labels are not only more interpretable, but may also be more flexible than the features learned by considering pre-computed molecular representations.

4 Problem Statement

Two gaps in the existing literature motivate this work. The first gap is the lack of studies that have comparatively evaluated the performance of different GNN architectures for the task of predicting molecular toxicity. While there are a variety of GNN models that have been applied to the task of predicting other molecular properties [19], [20], [21], [22], [23], few have evaluated different architectures for the task of toxicity prediction alone.

The second gap in the literature is with respect to the type of molecular representations that have been used within the different models. Each of the classes of

models (graph-based, fingerprint-based, and descriptor-based) encodes different aspects of the molecules' structures [13], and there is no theoretical reason why any one of those representations is expected to capture all of the aspects of those molecules that are relevant to the toxicity of those molecules. Each of these model types makes such a commitment to a single type of representation, but publishes their results without evaluating the gain (or loss) of incorporating such a restriction on the types of representations from which the models are built. The question of whether incorporating all three types of representations into a single modeling framework yields any improvement in accuracy over the best-performing individual approach for the LD_{50} endpoint is an open question.

5 Research Objectives

The work presented in this thesis pursues five objectives:

- To benchmark five different GNN architectures (GCN [24], GAT [25], GIN [26], GraphSAGE [27], and Graph Transformer [8]) on the LD_{50} dataset while controlling for all variables except for the neural network architecture.
- To conduct ablation studies that determine the impact of different graph-level components (readout, depth, batch normalization) to arrive at the validated best architecture.
- To evaluate the performance of two alternative model architectures based on molecular descriptors (2D CNN and MLP).
- To develop and evaluate the DeepToxiGraph model that integrates the graph, fingerprint, and descriptor models into a single model to determine the benefit of integrating these different representations of the molecules into the model in comparison to each of them individually.
- To develop a web application based on the DeepToxiGraph model that enables non-programmer end users to input molecular structures and obtain toxicity predictions

6 Scope of the Research

All experiments are performed on one endpoint: the intraperitoneal LD_{50} in mouse models from the TOXRIC database [11] containing 35,299 compounds. This endpoint was chosen due to its extensive use in pharmaceutical toxicity studies and its

availability in a public database. Other endpoints, such as hepatotoxicity, mutagenicity, or chronic toxicity, are outside the scope of this study.

The GNN evaluation covers five architectures. The fingerprint branch evaluates nine fingerprint types computed with the RDKit [28], and the descriptor branch uses 208 RDKit physicochemical descriptors [28], [1] for the standalone MLP baseline and 208 for the hybrid model. All models share the same random split of the dataset into 70 percent training and 30 percent test, with a fixed random seed of 42, and are evaluated using three metrics: R^2, RMSE, and Pearson r.

7 Thesis Outline

The remainder of this thesis is structured as follows.

Chapter One provides the general introduction to this thesis and outlines the overall context of the research. It presents the problem statement, defines the main objectives of the study, and discusses the scope and limitations of the work. In addition, it offers a structured overview of the thesis and concludes with a brief summary of the chapter.

Chapter Two builds the theoretical foundations of DeepToxiGraph. This chapter reviews the history of computational drug design, the four representations of molecules that are utilized in this thesis, the mathematics of graph theory and how it relates to molecules, the history of approaches to predicting the toxicity of drugs, and the neural network architectures that are employed in the experiments detailed in the following chapters.

Chapter Three details the experiment that was performed as part of this thesis. More specifically, Chapter Three describes the preparation of the dataset to be utilized in the experiment, the construction of molecular graphs from the SMILES notations of the molecules, and the descriptions of the six stages of the experiments performed.

Chapter Four describes the implementation and results of each of the six stages of the experiment, accompanied by tables and figures that illustrate the results of each stage. It also presents the web application developed to deploy the DeepToxiGraph model, enabling researchers to obtain LD_{50} predictions directly from a SMILES string without writing any code.

Finally, the Conclusion is the final chapter, which discusses and concludes this work, highlights the contribution and results of our work, and provides suggestions and recommendations for future research.

8 Summary

This chapter has established the context and direction of the DeepToxiGraph project. Beginning from the practical impossibility of experimentally characterizing every synthesizable compound, we traced the evolution of computational toxicity prediction and identified the representational limitation that constrains most existing QSTR methods: the loss of relational molecular structure at the encoding step.

Graph neural networks address this limitation directly, and DeepToxiGraph was motivated by the absence of a systematic, controlled study that evaluates GNN architectures specifically for LD_{50} prediction and that asks whether combining graph, fingerprint, and descriptor representations within a single model produces better predictions than any one of them alone. Five concrete objectives were stated, a precise scope was defined, and the structure of the remaining chapters was laid out. The theoretical tools needed to follow the experimental work are developed in Chapter Two, which covers the mathematical and conceptual foundations of every method applied in Chapters Three and Four.

CHAPTER 2

LITERATURE REVIEW

1 Introduction

This chapter presents the theoretical background required to understand the DeepToxiGraph framework for predicting molecular acute toxicity. We begin with the pharmaceutical context of drug design and computer-aided drug design, then describe the molecular representations used in this work: SMILES, molecular fingerprints, physicochemical descriptors, and molecular graphs, followed by the basic concepts and matrix-based formalisms of graph theory adjacency, degree, Laplacian, and incidence matrices that form the mathematical foundation of the GNN architectures. The toxicity prediction problem is introduced next, followed by a review of how computational methods have evolved from classical QSAR models, through multi-task deep neural networks, to graph neural networks. The mathematical foundations of machine learning, deep learning (convolutional neural networks, GNN architectures, and attention mechanisms), regularization techniques (batch normalization and dropout), and readout functions are then presented.

2 Drug Design

Drug design is a new field that is based on computing, and it is a process aimed at discovering new compounds and improving them so they can show specific biological activities. This process is based on the idea that molecules with similar structures tend to have similar properties and biological effects and activities. Based on this, the biological activity of a new compound can be predicted by comparing it to a database of known compounds that already have specific biological activities [29].

2.1 Computer-Aided Drug Design

Computer-aided drug design is the application of computational methods to the discovery and optimization of potential drug candidates [30]. Through the use of in silico methods to screen potential compounds for their biological activity, researchers are able to identify the most promising candidates for synthesis and testing in the laboratory, thus reducing the cost and time necessary to discover new drugs. CADD methods fall into two broad categories depending on whether the 3D structure of the biological target is known: structure-based and ligand-based [30].

2.1.1 Structure-Based Computer-Aided Drug Design (SB-CADD)

Structure-based computer-aided drug design is based on the 3D structure of the biological target [30]. The theory is that a protein will react biologically if a tiny molecule can fit into its binding site and interact with the protein molecule. Similar biological reactions will result from small molecules fitting into the same protein site in the same manner. This characteristic may be used to find tiny compounds with the appropriate biological activity [30].

2.1.2 Ligand-Based Computer-Aided Drug Design (LB-CADD)

Ligand-based computer-aided drug design is used if the three-dimensional structure of the target protein is not available. Instead of focusing on the target protein, LB-CADD considers a set of compounds that are known to bind to the target protein. The features that are shared by these active compounds are considered to create a model that indicates the features that are necessary to create active compounds. New compounds can be created or ranked based on how well they have these features [30].

3 Molecular Representation

For any computational model to process a molecule, the molecule must first be converted into a numerical format. This conversion step, known as molecular representation, has a decisive influence on the quality of the downstream models [13], [31]. Four main representations are employed in this work.

3.1 SMILES

The SMILES (Simplified Molecular Data Input System) encodes the molecule as a linear string of characters.[31] Each atom other than hydrogen is represented by its chemical symbol. Each single bond is implied, each double bond has the symbol "=", each triple bond has the symbol "#", aromatic systems use lowercase letters, and numbers represent how the molecule coils on itself to form. It is a global format for exchanging molecular data used in chemical databases around the world. SMILES is well-suited for sequence-based neural networks.

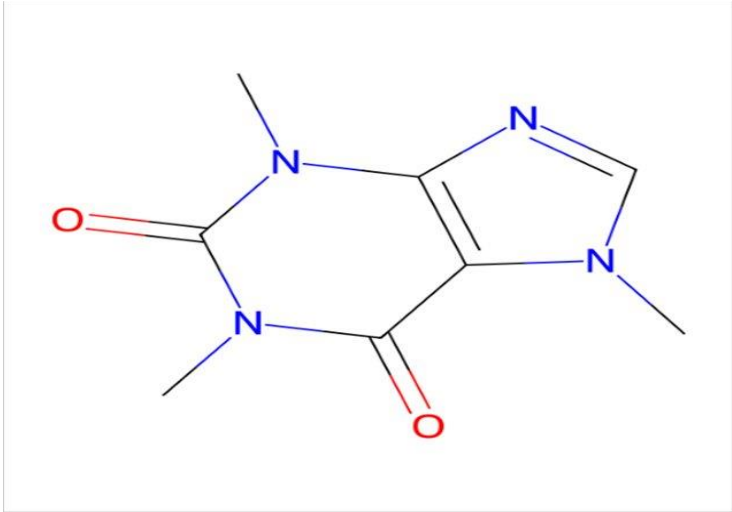


Figure 1: SMILES Representation.

3.2 Molecular Fingerprints

Molecular fingerprints are represented as binary or numerical vectors whose dimensions represent the presence or absence of certain substructures within the molecules [13]. By representing the structures of molecules in this way, their structures can be easily stored and processed by computers. These representations are often employed in tasks in machine learning and chemical informatics, such as finding similar molecules or grouping molecules [15], [32]. The main families are:

- Circular fingerprinting (ECFP): The chemical environment around each atom is encoded up to a specific bond radius using Morgan's algorithm [14]. This encoding represents the unique substructural features of a molecule in a number that is useful for similarity comparisons and models [13], [14].
- FCFP, Avalon [28].
- MACCS [28].

3.3 Physicochemical Descriptors

The physicochemical descriptors are the numerical values of the physicochemical properties of a molecule at different levels of molecular representation [1], These descriptors are essential in many scientific domains, such as pharmaceutical sciences and chemistry.

- The 0D descriptors represent the global properties of the molecule and are independent of the structural representation of the molecule [1].
- The 1D descriptors are based solely on the molecular formula of the molecule [1].
- The 2D descriptors are calculated from the two-dimensional representation of the molecule's connectivity [1].
- The 3D descriptors are calculated from the three-dimensional representation of the molecule [1].

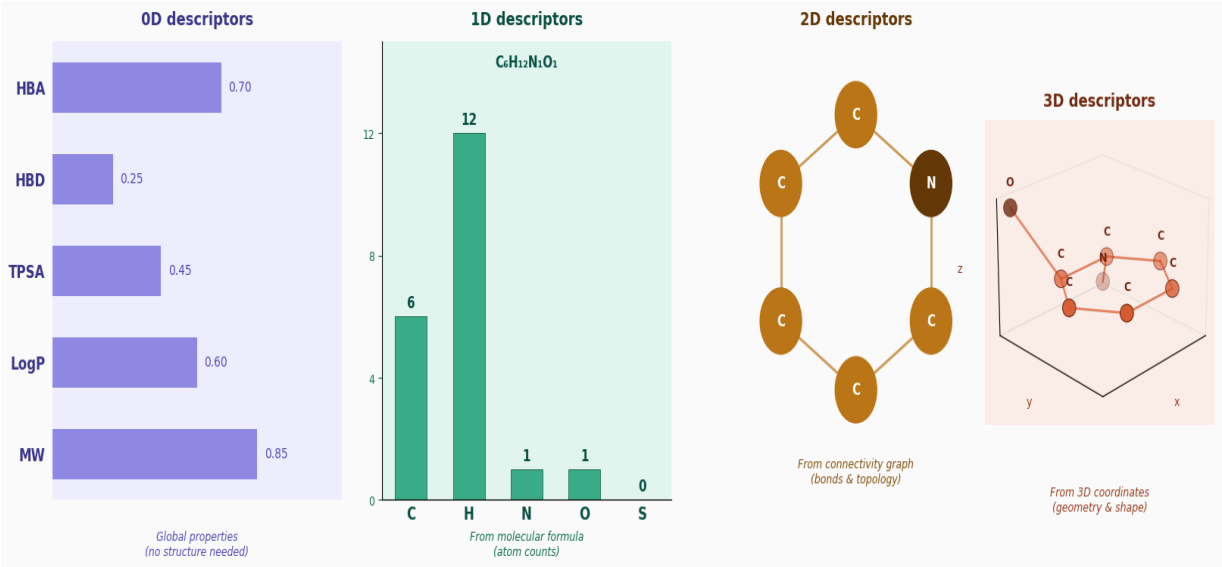


Figure 2: Molecular descriptor representations RDKit [1].

The RDKit library provides implementations of more than 200 physicochemical descriptors [28].

3.4 Molecular Graphs

In a molecular graph, the atoms of the molecule represent the nodes of the graph V, while the chemical bonds between these atoms form the edges of the graph E [13], [6]. Each node v of the graph is associated with a feature vector xv, which encodes various properties of the atom at that node. These properties include the atomic number, formal charge, oxidation state, orbital hybridization type, aromatic membership, number of rings containing that atom, and number of radical electrons. Each edge (u, v) of the graph is associated with a feature vector eu,v, which encodes the bond type between the two atoms u and v, ring membership, length, and rotation [13], [6].

The molecular graph is the richest of the four representations, as it retains all of the information regarding the structure of the molecule without loss of that information. It is the representation used by all of the GNN models evaluated in this work.

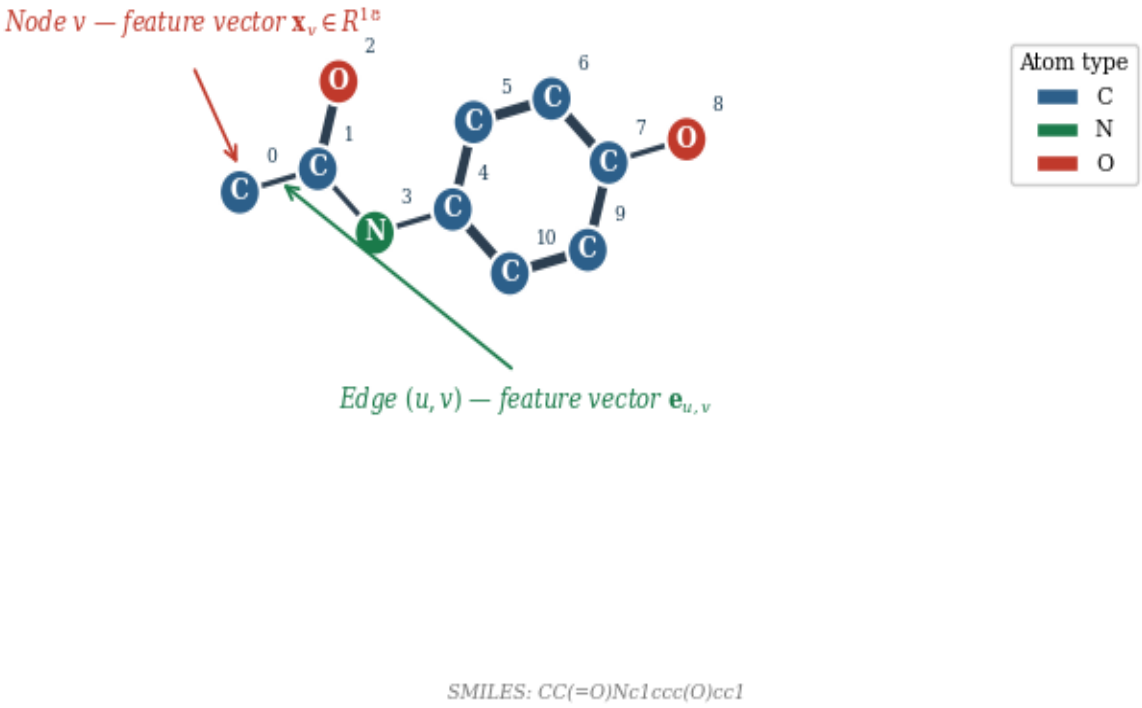


Figure 3: Molecular Graph.

4 Graph-Theoretic Representations

A graph G = (V, E) is defined as a set of two sets, V and E, where V is a finite set of nodes, and E ⊆ V × V is a set of edges [2]. Nodes (vertices), are the entities or data points in a graph and they are the atoms in a molecule. The edges are the links between the nodes and represent the relationships between them. They are either directed or undirected [2].

Many characteristics can be used to classify graphs. If a graph has a finite number of vertices and edges then it is finite graph and if not then it is undirected graph . Each edge in a directed graph has a direction specified, but an undirected graph does not [2].

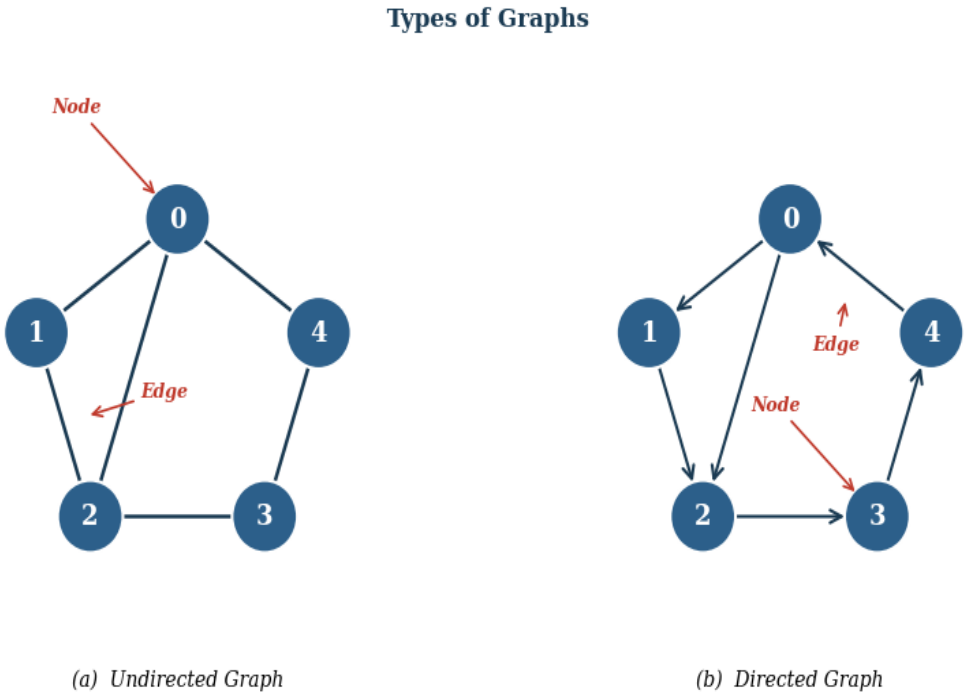


Figure 4: Directed and undirected graphs [2].

4.1 Adjacency Matrix

The simple graph has its corresponding adjacency matrix denoted by $A \in \{0,1\}^{N\times N}$. The element Aij represents the connection between node vi and node vj.:

$A_{ij} = 1$ if v_i is adjacent to v_j and $i \neq j$,

and $A_{ij} = 0$ otherwise [3]. (2.1)

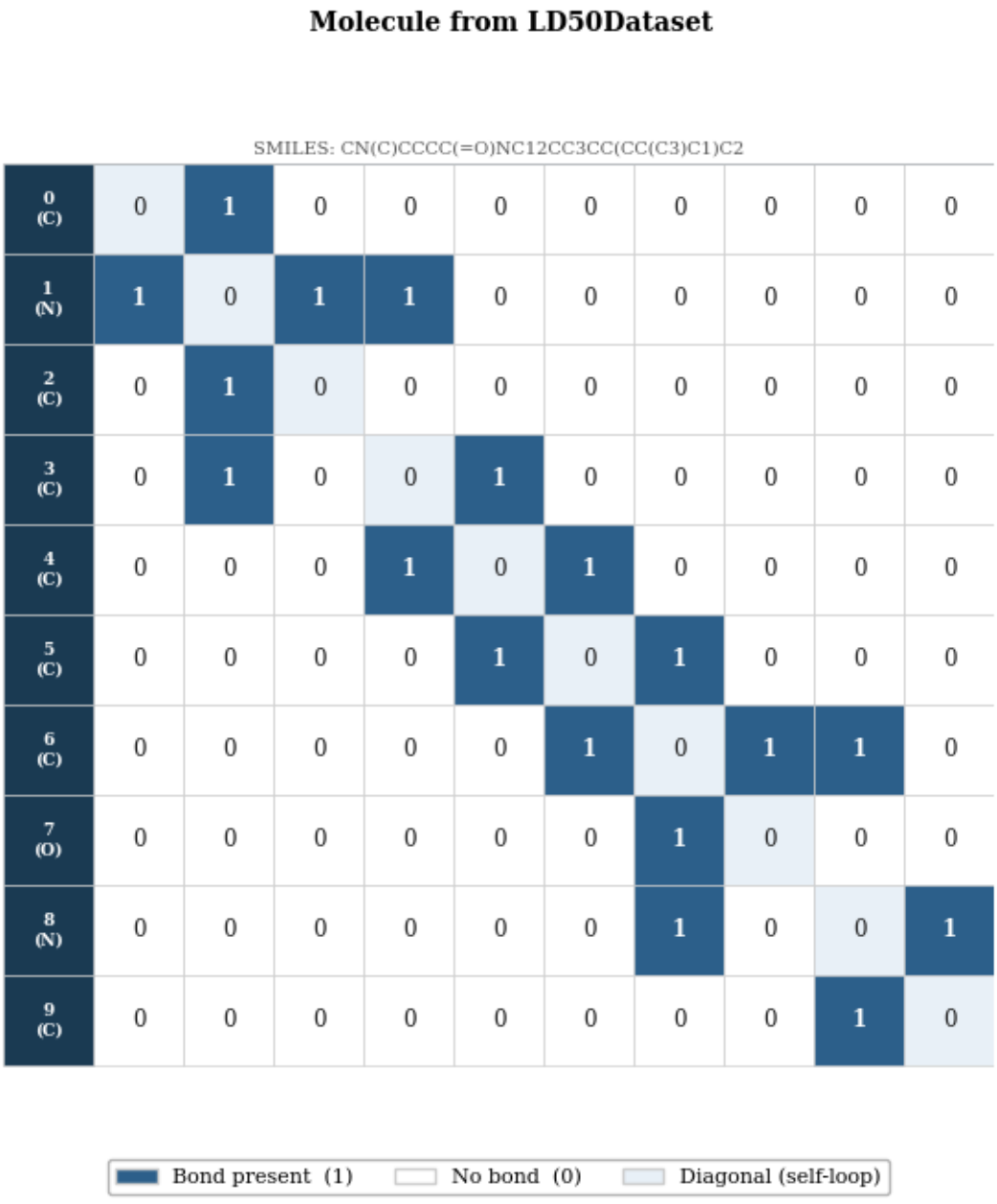


Figure 5: Adjacency matrix [3].

4.2 Degree Matrix

A graph having n edges, the degree matrix $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix where the diagonal entry Dii corresponds to the degree d(vi) of vertex vi [3].

4.3 Laplacian Matrix

The Laplacian matrix $L \in \mathbb{R}^{n \times n}$ of a simple undirected graph is defined as follows:

L = D − A, where D is the degree matrix and A is the adjacency matrix and:

$L_ij = d(v_i)$ if $i = j$,

$L_ij = -1$ if $\{v_i, v_j\} \in E$ and $i \neq j$,

$L_ij = 0$ otherwise [3]. (2.2)

4.4 Symmetric Normalised Laplacian

Lsym = D−1/2 L D−1/2 = I − D−1/2 A D−1/2 is the definition of the symmetric normalised Laplacian Lsym. It consists of [3]:

$L^{sym}_{ij} = 1$ if $i = j$ and $d(v_i) \neq 0$,

$-1 / \sqrt{(d(v_i)\, d(v_j))}$ if $\{v_i, v_j\} \in E$ and $i \neq j$,

and 0 otherwise. (2.3)

4.5 Incidence Matrix

The incidence matrix $M \in \mathbb{R}^{n \times m}$ is defined for a simple graph with n nodes and m edges so that :

$M_{ij} = 1$ if edge e_j is incident to node v_i,

and 0 otherwise [3]. (2.4)

5 Toxicity Prediction

Animals or cells must be tested in lab in order to evaluate a chemical compound's safety. These tests are expensive, time-consuming, and pose ethical questions about using biological organisms for testing. By creating a mathematical mapping between a chemical compound's structure and its measured toxicity, computational toxicity prediction seeks to evaluate novel compounds without the requirement for experimental testing [18].

Toxicity refers to a family of related but distinct properties of compounds that are determined experimentally. The LD_{50} of a compound is one of the most difficult of these properties to model. Lethality can be the result of a variety of unrelated biochemical processes occurring within the body simultaneously [10]. However, because there exist large publicly available datasets of compounds and their LD_{50} values, this property is one of the most suitable for evaluating different GNN architectures [10].

6 Prior Research on Toxicity Prediction

The computational prediction of molecular toxicity from structural information has evolved through successive generations of methods.

6.1 Classical QSAR Models

Quantitative structure-activity relationship (QSAR) model creates a mathematical connection between a molecule's activity and structural characteristics [12]. According to QSAR models, molecules with comparable structures will behave similarly. A QSAR model is constructed in the following steps [12], [17]:

- Molecular structure preparation.
- Dataset splitting for training set and testing set.
- Calculation and selection of molecular descriptors.
- Construction of the QSAR model utilizing the selected molecular descriptors.
- Validation of the model by testing it on a test set of molecules.

Early QSAR models included principal component analysis, partial least squares, and multiple linear regression using manually created descriptors [12]. These models did well in chemically homogeneous series, but they did not generalize well to datasets with different structural characteristics. One of the first LD_{50} models on big datasets was published by Zhu et al. [10], who found that the key restriction was poor out-of-domain generalization.

6.2 Deep Neural Networks

Deep neural networks were also employed in extending QSAR modelling to include non-linear relationships between the descriptors and the biological activity [18]. Mayr et al. [18] developed DeepTox, a multi-task deep neural network trained on ECFP fingerprints and a large set of chemical descriptors for the Tox21 challenge. DeepTox predicted twelve toxicity endpoints simultaneously and won the challenge, reporting a mean test-set AUC of approximately 0.846 across the twelve tasks and outperforming naïve Bayes, support vector machines, and random forests. Stokes et al. [19] applied a directed message-passing graph neural network to antibiotic discovery; trained on a binary growth-inhibition dataset of 2,335 molecules, their model reached a test-set ROC-

AUC of about 0.896 and led to the identification of halicin, a novel antibacterial compound effective against drug-resistant bacteria. Duvenaud et al. [20] introduced differentiable “neural graph fingerprints” that replaced fixed circular fingerprints with a graph convolutional network learned end-to-end, improving over fixed ECFP fingerprints on solubility, drug efficacy, and photovoltaic-efficiency regression benchmarks. Yang et al. [21] developed the directed message-passing neural network (D-MPNN, released as Chemprop) that learns molecular representations directly from the graph; across a wide range of public and proprietary datasets it consistently matched or outperformed models based on pre-computed molecular fingerprints, establishing message passing as a strong default for molecular property prediction.

A smaller body of work targets the acute-toxicity LD_{50} endpoint specifically, which is the regression task addressed in this thesis. Zhu et al. [10] built one of the first large-scale rat acute oral LD_{50} QSAR models, using a dataset of roughly 7,400 compounds compiled from the ChemIDplus and admetSAR sources, trained with k-nearest neighbours, support vector machines, and other classical learners on calculated descriptors, while highlighting poor generalization outside the training domain as the central limitation. Lu et al. [33] proposed a local lazy learning approach that builds a local model around each query compound from its nearest neighbours, evaluated on a dataset of 7,385 rat acute oral LD_{50} compounds. Xu et al. [34] introduced deepAOT, a molecular-graph-encoding convolutional neural network for acute oral toxicity, trained on the acute oral toxicity dataset derived from the admetSAR database. Most relevant to this thesis, Saravanan et al. [35] used the same TOXRIC mouse intraperitoneal LD_{50} dataset employed in this work and applied a graph convolutional network regression model to predict the LD_{50} value directly from molecular structure, as part of a broader multi-model toxicity-prediction framework. These studies differ in the dataset, species, and route of administration they use, which is why a direct comparison of their performance is only meaningful when the same dataset is used; such a comparison, restricted to the TOXRIC mouse intraperitoneal LD_{50} dataset employed in this thesis, is presented in Chapter 4.

7 Machine Learning

Machine Learning is a method for obtaining Artificial Intelligence [4]. It focuses on creating algorithms that allow computers to learn from data and make decisions or predictions based on that data [4]. Some areas in which machine learning is applied

include computer vision, speech recognition, recommendation systems, and natural language processing [4].

ML can be:

- Supervised learning: using labeled datasets [4]. It is utilized for classification and regression applications [4].
- Unsupervised learning: looks for patterns in unlabeled data to categorize them [4].
- Reinforcement learning: based on learning through rewards and penalties [4].

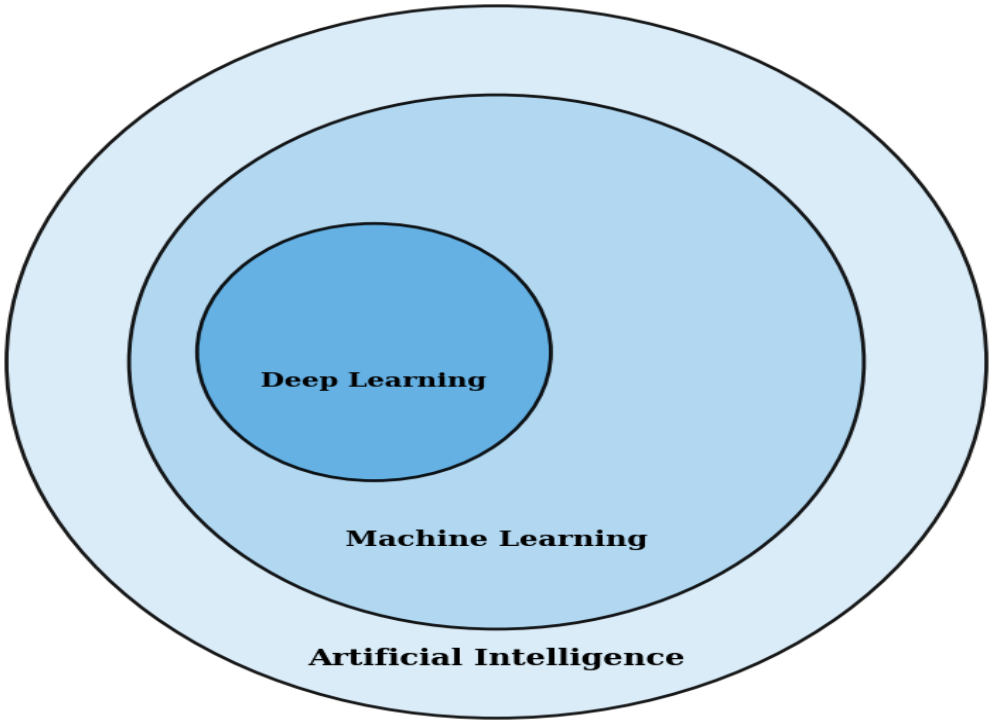


Figure 6: Deep learning is a subfield of machine learning, which is in turn a subfield of artificial intelligence [4].

8 Deep Learning

Deep learning is a subfield of machine learning that uses artificial neural networks with many stacked layers of non-linear transformations [36], [37], [38]. Such deep neural networks learn increasingly abstract features from the data so that complex features within the data can be represented in the deep learning model.

Deep learning models perform end-to-end training via backpropagation algorithms that compute the gradient of the loss function for the model's parameters [37].

8.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a class of deep neural networks that learns patterns from data that has a grid structure to it, such as images [5]. Instead of connecting each neuron in one layer to each neuron in the next layer in a neural network, a CNN learns a small number of weights (filters or kernels) that are applied across the entire input layer. The same set of weights is applied to every location within that input layer, taking advantage of the fact that many of the same patterns exist throughout the input data.

A typical CNN is built with three types of layers. Convolutional layers learn the weights of a filter that is applied to the input data to highlight regions of interest within the input. Pooling layers (typically max-pooling operations) reduce the size of the feature maps created by the convolutional layers to provide translation invariance to the CNN. Finally, fully connected layers make predictions based on the features that were detected by the convolutional layers [5]. The architecture of a typical CNN is represented in the figure below.

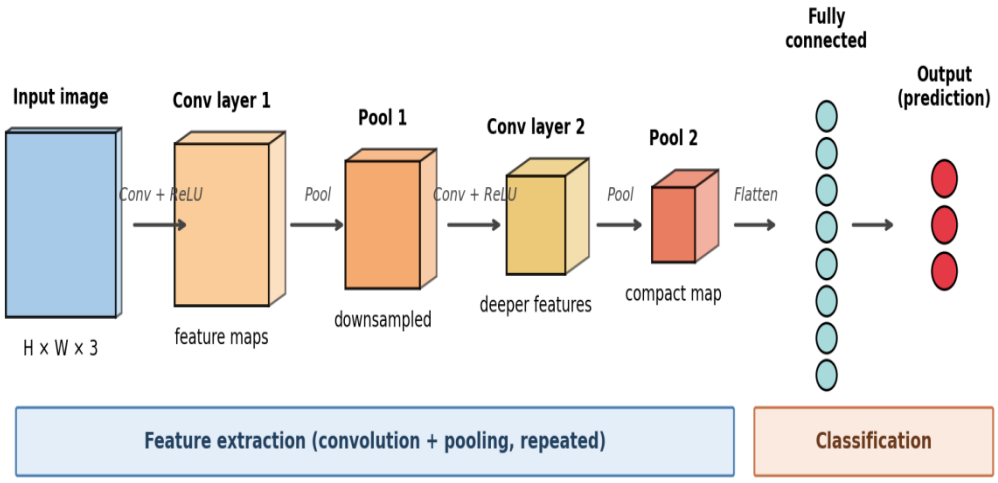


Figure 7: Typical CNN architecture [5].

Convolutional layers can be understood by considering the filter that is applied to the input data. For a two-dimensional convolutional layer, a 3×3 filter is applied to the input data such that each of the weights within the 3×3 filter is multiplied by each of the pixels within that same 3×3 region of the input image. The resulting values are summed to form a single scalar value, which is passed through an activation function to emphasize only those features that are important to the recognition of patterns within the image. The next figure illustrates how these two-dimensional convolutional layers apply the 3×3 filter to the input image to highlight certain features within the image.

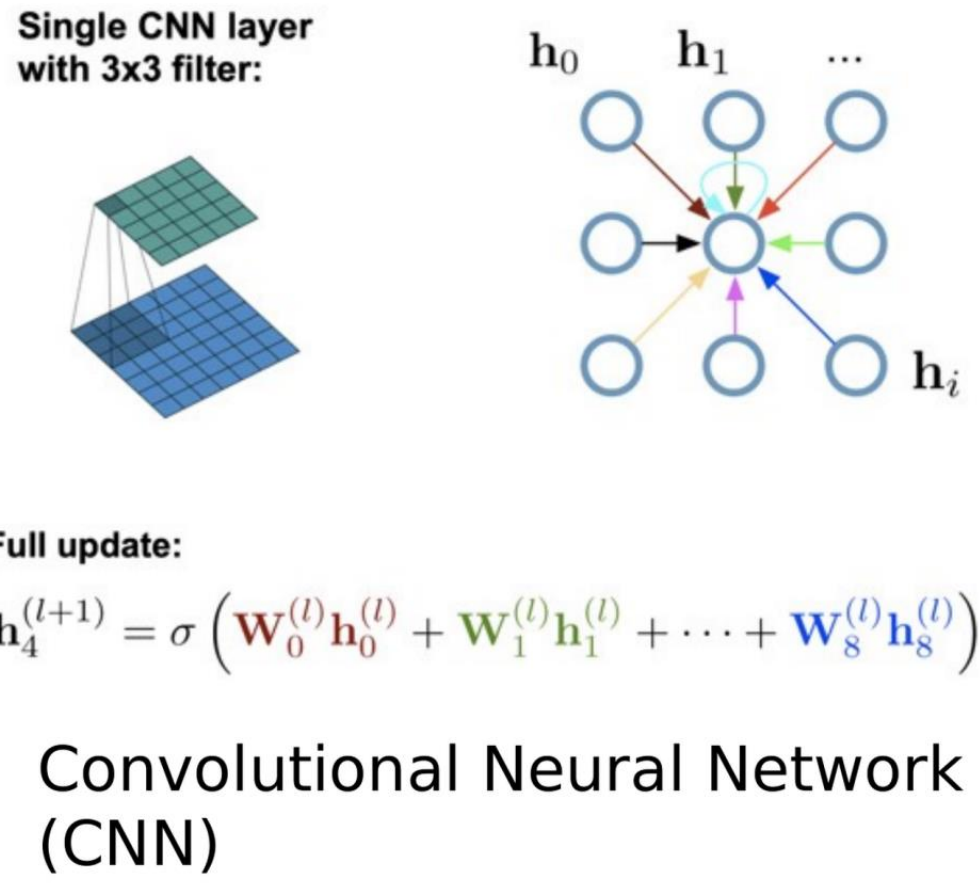


Figure 8: 3×3 convolutional filter [5].

For a feature h at position 4 (the centre of a 3×3 window of pixels indexed from 0 to 8) at layer ℓ+1:

$$h_4^{(\ell+1)} = \sigma(W_0^{(\ell)} \cdot h_0^{(\ell)} + W_1^{(\ell)} \cdot h_1^{(\ell)} + \ldots + W_8^{(\ell)} \cdot h_8^{(\ell)}) \quad (2.5)$$

The use of the same weight matrix W^(ℓ) across the entire image makes the network translation equivariant. While this has proven to be a very useful inductive bias in the case of images, it prevents the use of CNNs in the case of molecules. Molecules do not possess the same structure as a grid of pixels. The number of atoms in a molecule is variable, there is no inherent ordering to the atoms within a molecule, and the bonds between the atoms are not of the same form as the adjacency of pixels in a grid.

8.2 Graph Neural Networks

A Graph Neural Network (GNN) is a deep learning model whose input is a graph and whose output transforms in a predictable way when the nodes of that graph are reordered [39], [40], [41], [42], [7]. A graph G has a set of nodes (V) and edges (E) that join the nodes. The number of nodes in a graph is variable, as is the number of edges connected to each node. A GNN is a model that is generalizable to graphs of any size and whose output is the same regardless of the order in which the nodes of that graph are examined.

For molecules, each atom is represented as a node in the graph, and each bond between atoms is represented as an edge. Each node in the graph is associated with a feature vector for that atom. Each feature is a description of the atom; it can be elemental (what type of element is represented by the atom), its state of hybridization, its formal charge, whether it is present in a ring structure in the molecule, and other features. Each feature has a value associated with it; binary features have a 0 or 1 value, while other features are represented as one-hot vectors. These feature vectors are associated with each node (atom) in the graph, resulting in the initial embedding for the graph of molecules (shown in the next figure).

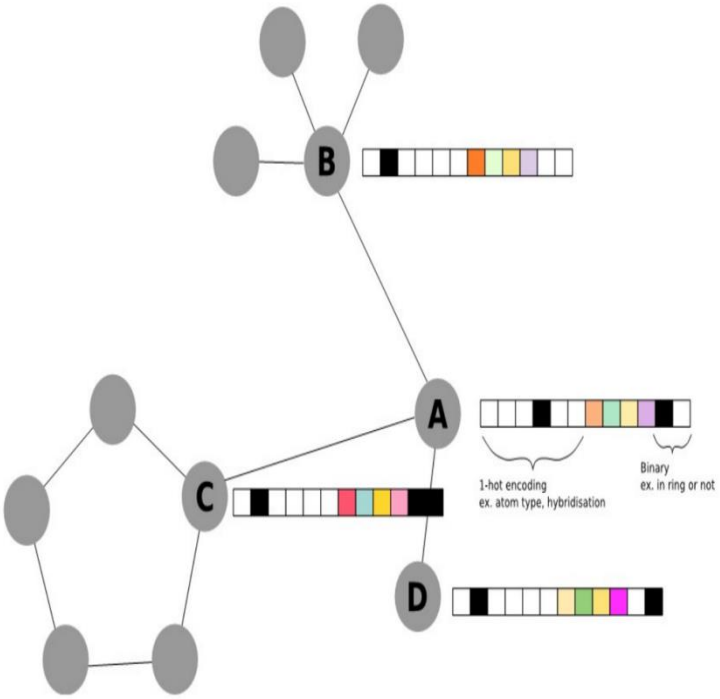


Figure 9: Atom feature encoding [6].

GNNs can be thought of as a generalization of CNNs [7]. In a CNN, a filter is applied to an image that includes the image pixel at the center of the filter with the pixels in the image that are neighbors of that center pixel (as shown on the left in the next figure). In a GNN, a node is mixed with its neighbors in the graph (as shown on the right in the next figure). The CNN is a special case in which the graph that is being examined is a grid, in which each point has the same neighbors; the GNN, by contrast, removes the requirement that those neighbors be regular in any way.

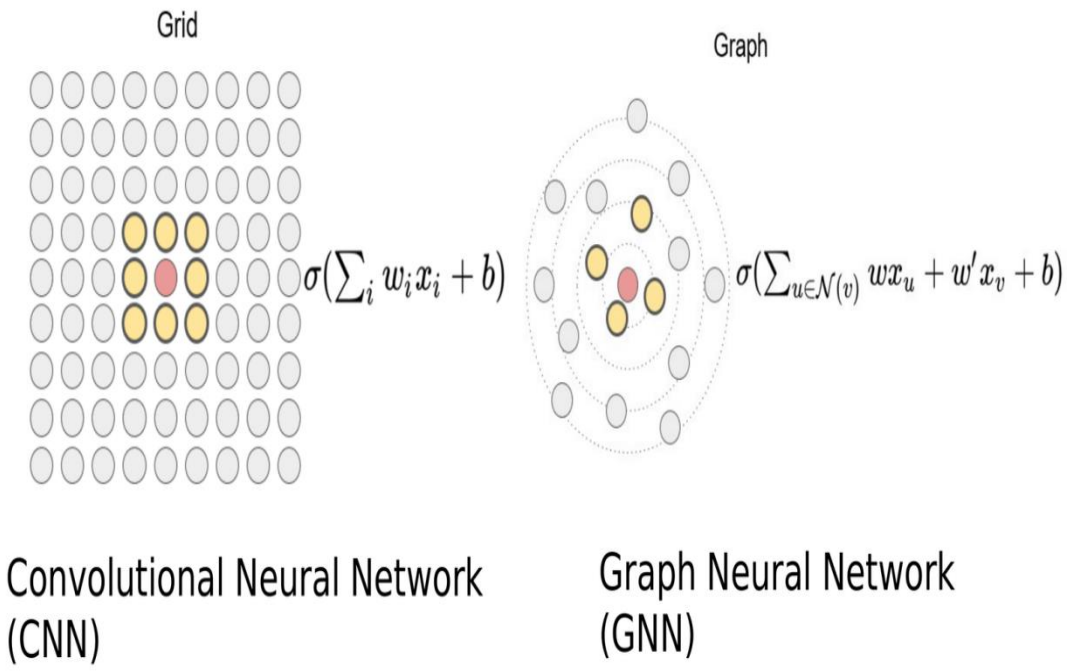


Figure 10: CNN versus GNN [7].

The same distinction can be made in the update rule of the CNN and the GNN. The CNN mixes the value of the pixels in the 3×3 patch of the image with the center pixel to create the new value of the center pixel. Similarly, the GNN mixes the values of the neighbors of a node in the graph with the value of the node itself to form a new value for that node. The weights applied to the neighbors in the graph are the same for each neighbor. The GNN is insensitive to the order in which the atoms of the molecule are listed.

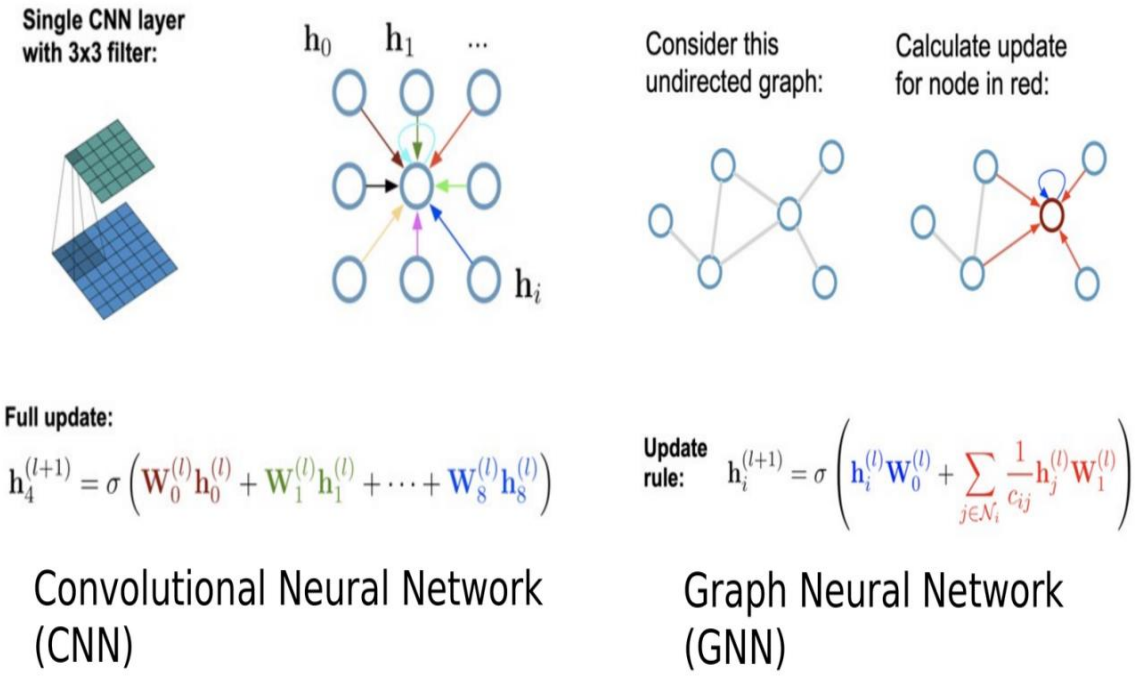


Figure 11: CNN and GNN updates [7].

8.2.1 Graph Convolutional Network (GCN)

The Graph Convolutional Network [24] is one of the first GNNs to be introduced and is among the most widely used. Each node's new representation within a GCN layer is the average of the old representations of the nodes that neighbor it (including itself), with the averaging weighted according to the degree of the nodes that connect to each other [7]. The representations of the neighbor nodes are multiplied by a shared matrix W^(ℓ) prior to taking the average to provide the layer with some of its learnable capacity. The result can be defined as follows:

$h_v^{(\ell+1)} = \sigma(\Sigma_{u \in N(v) \cup \{v\}} (1/c_{v,u}) \cdot W^{(\ell)} \cdot h_u^{(\ell)})$ (2.6)

where N(v) represents the neighbors of node v, and c_{v,u} is a normalization factor derived from the degrees of v and u. Despite the simplicity of this model, models with as few as three layers have been able to learn useful representations of the molecules, allowing them to make strong predictions on a variety of different data sets related to molecular properties.

In matrix form, the equation for a single layer of a GCN is F' = (I + A) · F · W, where F is the matrix of features of the atoms of size N × d_in, I is the N × N identity matrix, A is

the adjacency matrix of the molecule, and W is a learnable matrix of size d_in × d_out. The figure below represents three layers of a GCN being applied to a molecular graph of 93 atoms. The feature dimension d of each atom increases from 5 to 32, 64, and 128 as the depth of the network increases to 3 layers.

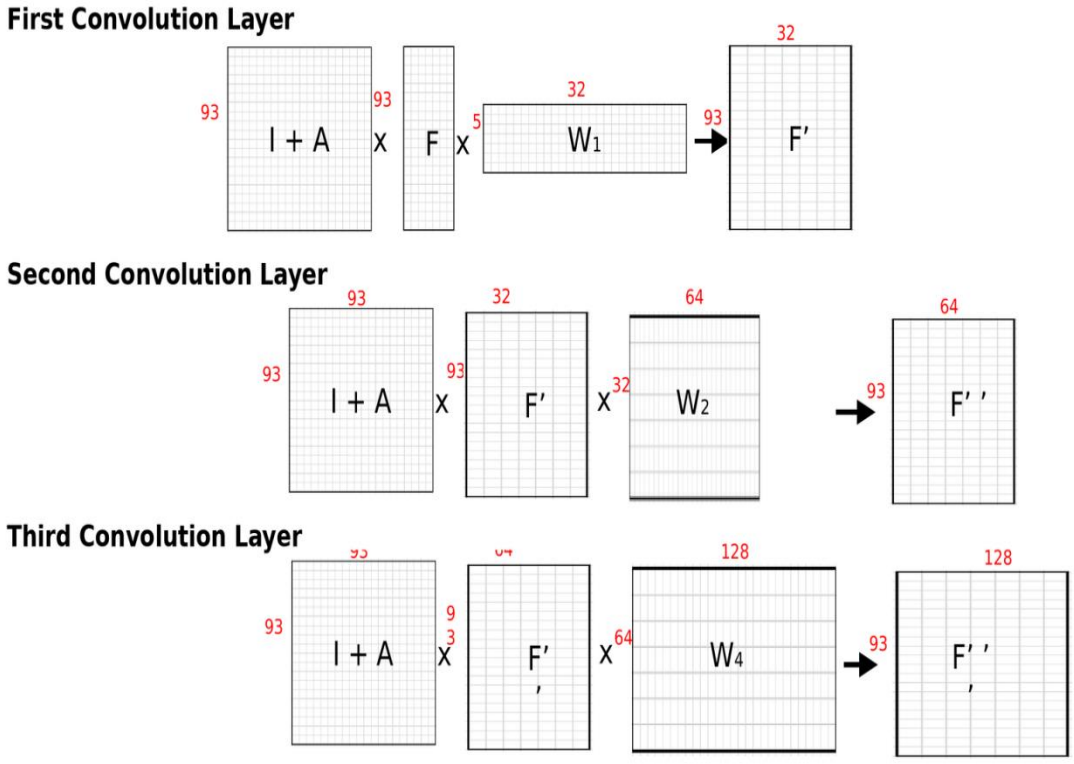


Figure 12: Stacked GCN layers [7].

8.2.2 GraphSAGE

GraphSAGE models node embeddings by aggregating features from neighboring nodes [27]. The representation of a node is updated using the representation of that node and the aggregated representations of its neighbors [27]. The update rule is defined mathematically as follows:

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, AGGREGATE_k(\{h_u^{k-1}, u \in N(v)\}))) \quad (2.7)$$

where:

- **h_v^k:** representation (embedding) of node v at layer k
- **h_v^{k−1}:** representation of node v from the previous layer

- **h_u^{k−1}:** representation of neighboring node u from the previous layer
- **N(v):** set of neighboring nodes of node v
- **AGGREGATE:** aggregation function used to combine neighbor features
- **CONCAT:** concatenation operation
- **W^k:** learnable weight matrix at layer k
- **σ:** nonlinear activation function

8.2.3 Graph Attention Network (GAT)

Both the GCN and GraphSAGE models treat each of the neighbor nodes of a given node as equally important up to a normalization factor. The Graph Attention Network (GAT) framework introduces a method of applying learnable attention weights α_{vu} to the features of each node's neighbors [25]. These weights are computed from the current embeddings by a small neural network and normalized with a softmax, giving the weighted-sum update:

$m_v = \Sigma_{u \in N(v)} \alpha_{vu} \cdot W \cdot h_u$ (2.8)

8.2.4 Graph Isomorphism Network (GIN)

The Graph Isomorphism Network also enhances the node representations by considering the characteristics of the neighbouring nodes, obtaining significant discriminative power similar to the Weisfeiler-Lehman test [26].

8.2.5 Graph Transformer

The Graph Transformer [8] applies the self-attention model that was presented in the original Transformer [9] to graph-structured data. The motivation for this approach can be expressed very concisely: in a standard GNN, information is propagated between nodes along the edges that comprise the graph; alternatively, a Transformer model can be thought of as a GNN in which the connectivity matrix is replaced with a connectivity matrix that is learned from the data. Combining these two approaches creates a model that can apply the inductive bias of the molecular graph to the representation learning process while also being able to up- or down-weight the importance of the connections between neighboring nodes according to the chemical context of those neighbors, as illustrated below:

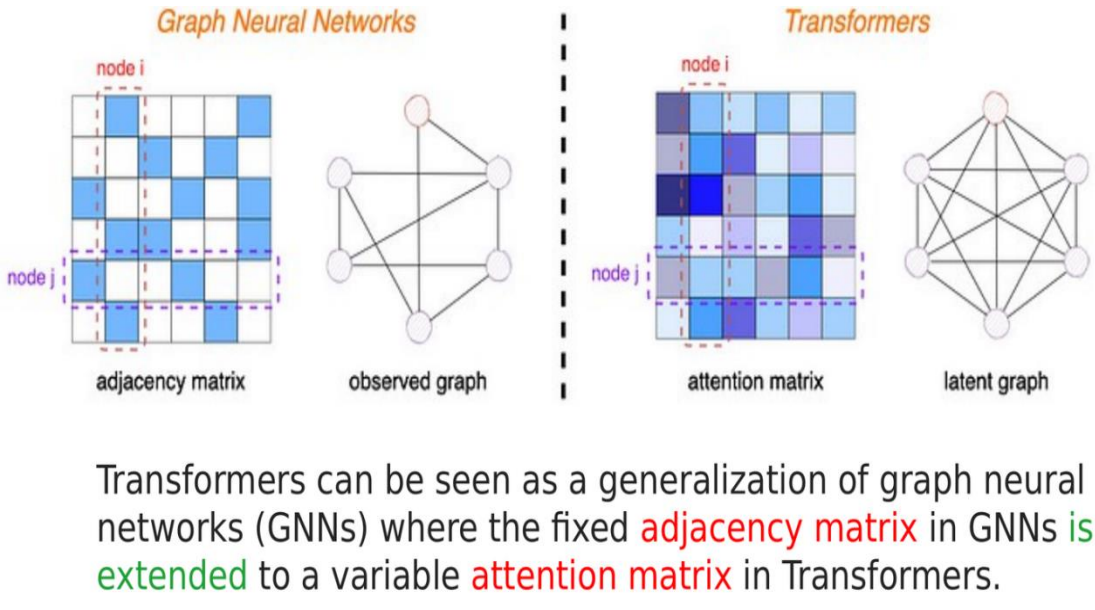


Figure 13: GNN versus Transformer [8].

For each node v in the graph G, and for each neighbor u of v, the Graph Transformer applies three separate linear transformations to the feature vectors of each node to create query, key, and value vectors for that node and neighbor:

$$Q_v = W_Q \cdot h_v\,, \quad K_u = W_K \cdot h_u\,, \quad V_u = W_V \cdot h_u + W_E \cdot e_{uv} \qquad (2.9)$$

The value vectors for each neighbor u of node v include the features of the edge that connects those two nodes through a separate linear transformation W_E. The importance of each neighbor u is computed as a scaled dot product between the query and key vectors Q_v and K_u, respectively, which is passed through a softmax function. The new feature representation for node v is the weighted sum of the value vectors V_u of its neighbors. The definition of this dot product-based attention function is presented in the next section

9 Transformers and Attention Mechanisms

The Transformer architecture was introduced by Vaswani et al. [9] as an architecture for modeling sequences of arbitrary length that did not rely on the use of recurrent connections between the different processing steps of the model. Recurrent models tend to be limited by the length of the sequences that they can effectively process due to the vanishing gradient problem. Instead, the Transformer model replaces the use of recurrence with what is known as self-attention, wherein each element of the sequence is

allowed to examine each of the other elements of the sequence and to learn which elements of the sequence are relevant to each element.

This type of mechanism can be understood through examining two different examples of its use. In the sentence “Understanding the protein-ligand interaction is key to drug discovery," for instance, the word "interaction" is related to the words "protein-ligand"; the attention layer can learn to recognize these relationships without examining the sentence element by element. Similarly, within a molecule, the same type of self-attention layer can be used to understand whether two fragments of the molecule, such as an -NH_2 amine group and a tert-butyl group, are located within a certain distance of one another within the molecule.

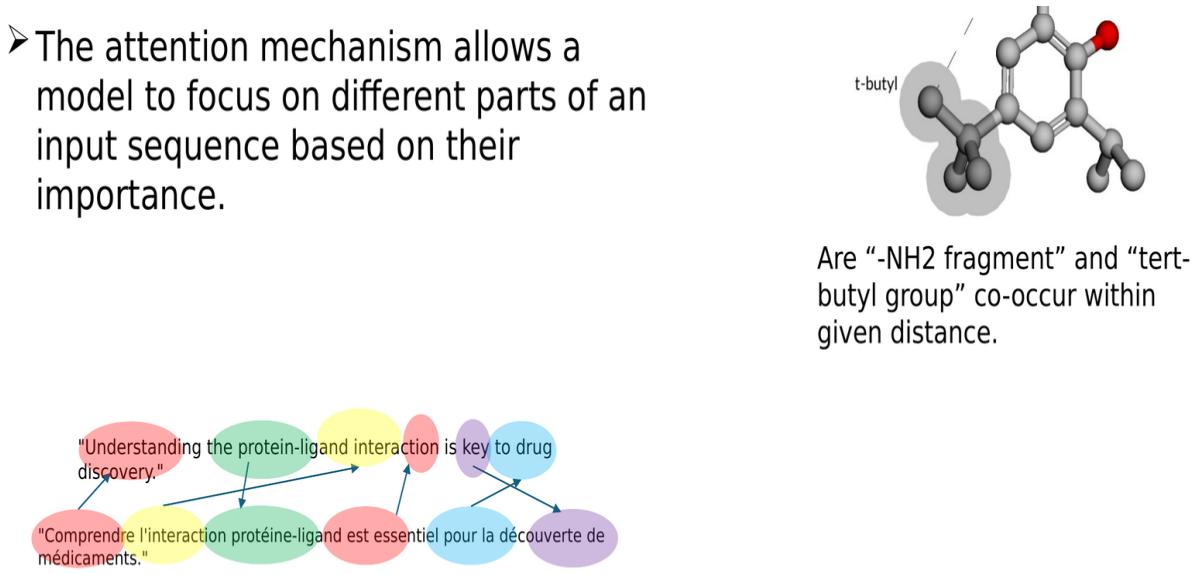


Figure 14: Attention mechanism motivation [9].

In the context of molecules, such an architecture is helpful in recognizing that the relationship between two atoms is not always represented by the presence of a direct bond between those two atoms. Instead, two groups located several bonds apart from each other may interact with one another, and the ability of the model to recognize these relationships allows it to function with an advantage relative to architectures that must examine the sequence of atoms in the molecule in order to determine if any interaction between those remote atoms occurs. The Graph Transformer that will be utilized in this work is, essentially, the use of the self-attention model for molecular graphs [8].

9.1 Scaled Dot-Product Attention

Self-attention uses three pieces of information that come from the same input. Given an input matrix X, three separate matrices can be created by performing multiplications:

$$Q = X \cdot W^{\wedge}Q\,, \quad K = X \cdot W^{\wedge}K\,, \quad V = X \cdot W^{\wedge}V \qquad (2.10)$$

In this equation, Q represents the information that each element is looking for. K represents the information that each element can offer. V represents the information that each element will send out when it finds what it is looking for. To determine how much each element should pay attention to another element, each element's Q matrix is multiplied by every other element's K matrix. The larger the result of that multiplication, the more information that element contains that can answer the question that the element is asking so its V values will be sent out.

These dot products are scaled by dividing each value by the square root of the dimension of the key matrix (d_k) and converted to percentages using the softmax function:

$$Attention(Q, K, V) = softmax(\, Q \cdot K^{T} / \sqrt{d_k}\,) \cdot V \qquad (2.11)$$

The reason that the value of the square root of d_k is used in the attention process is to ensure that the learning process works well for the system. Without this division of each dot product by the square root of d_k, the values of d_k will have a major influence on the values of the dot products. If d_k is a large dimension, the dot products will be large, leading to extreme values from the softmax function. These extreme values will make it difficult for the machine to learn. By dividing each dot product by the square root of d_k, the values are scaled to ensure that the learning process works well for the machine learning system.

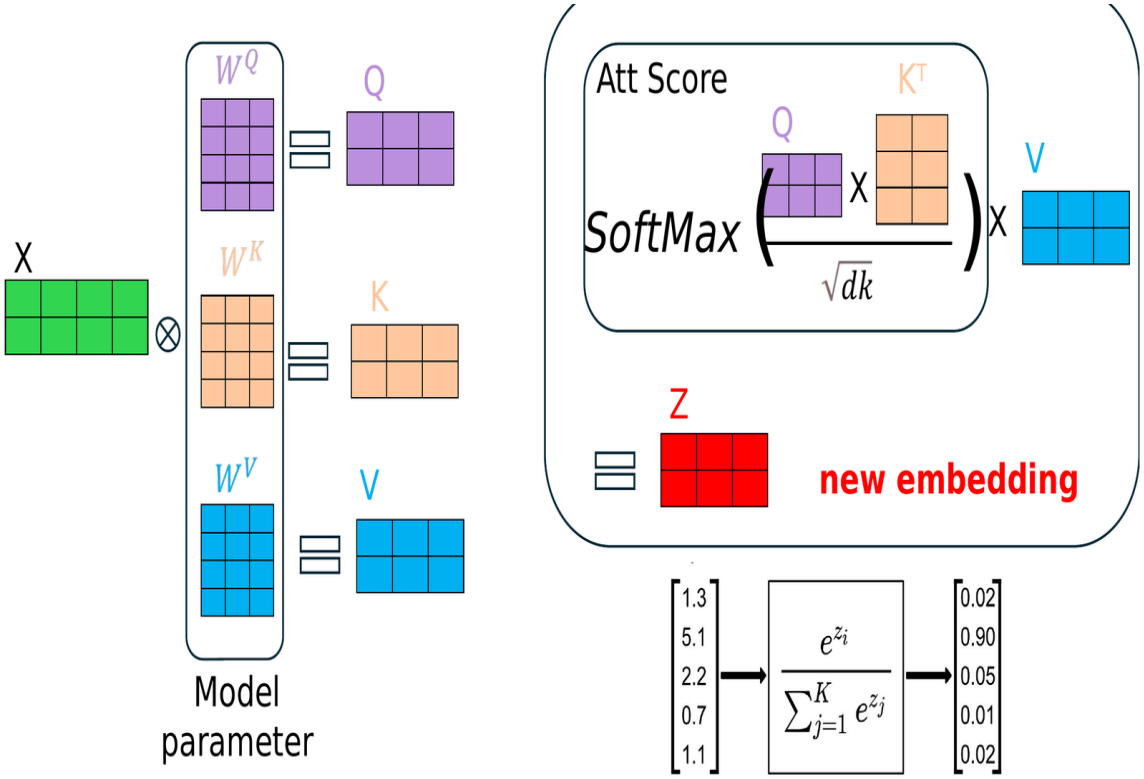


Figure 15: Scaled dot-product attention [9].

The figure represents the steps in the self-attention process. Each element of the input matrix X is multiplied by the weight matrices W^Q, W^K, and W^V to create the Q, K, and V matrices for each element. The query (Q) is multiplied by the keys (K) to find dot products. These dot products are scaled and normalized using the softmax function. The attention weights are used to create a weighted sum of all the V matrices to form the output matrix Z.

9.2 Multi-Head Attention

A single self-attention layer can only examine a single type of relationship between the elements within the matrix. It is often more useful to examine the same elements of a matrix from multiple different angles at once. Multi-head attention performs this by applying multiple self-attention operations simultaneously to the same inputs:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \cdot W^O \quad (2.12)$$

$$head_i = Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V) \quad (2.13)$$

Each of the heads within the multi-head attention layer has its own set of three projection matrices. Each of these heads can examine a different type of similarity between the elements of the matrix. A single projection matrix combines the results of

each of these heads into a single output. In the Graph Transformer used in this work [8], the attention is additionally restricted to the actual graph neighbourhood, so each head learns a different way of weighting neighbours rather than the whole molecule.

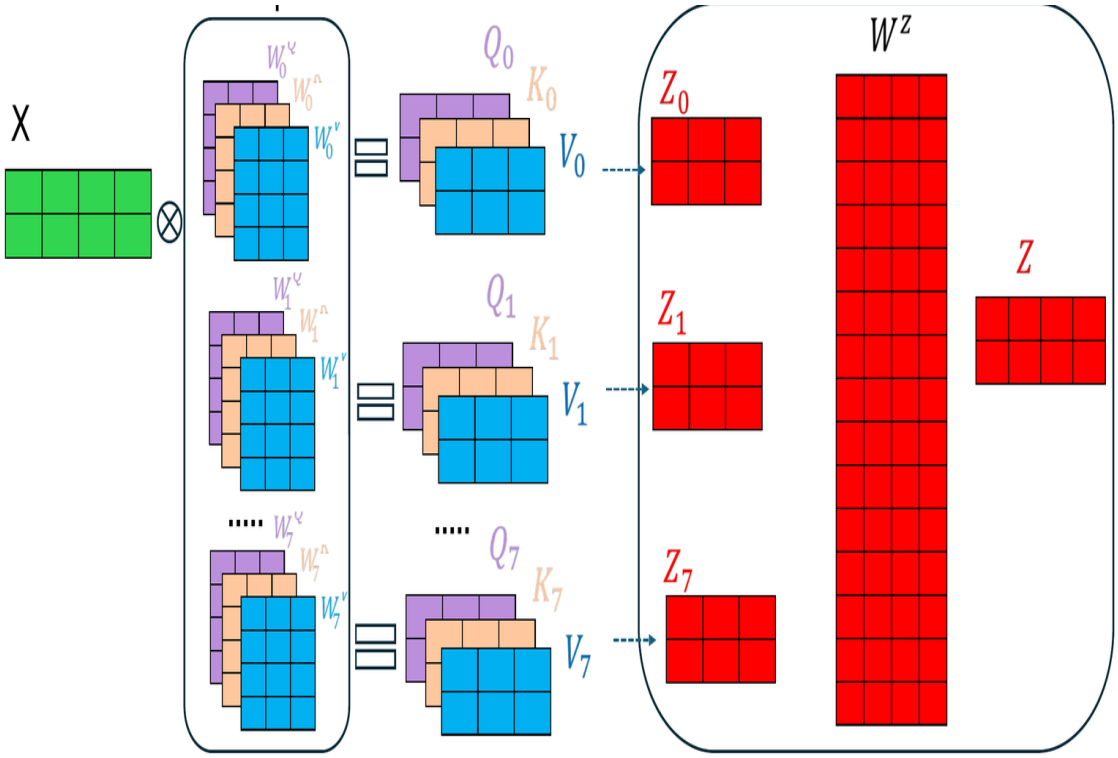


Figure 16: Multi-head self-attention [9].

Multiple heads are one of the main ingredients that make Transformers and Graph Transformers expressive relative to earlier models. They allow the model to consider several different types of interactions between atoms in a molecule at once and to combine those interactions appropriately into the representation of the molecule that is passed along to the readout and prediction layers in the following section.

10 Batch Normalization

Another training method for normalizing each layer's activation before moving on to the next layer is batch normalization [43]. The batch normalization transformation is defined as follows for each tiny batch of activations:

$$BN(x) = \gamma \cdot (x - \mu_B) / \sqrt{(\sigma_B^2 + \varepsilon)} + \beta . \qquad (2.14)$$

11 Dropout

Dropout is a regularization strategy that randomly removes neurons during training in order to minimize overfitting [44].

12 Readout Functions

After the message-passing layers, each atom in the molecule has its own vector [6]. A readout function combines all these atom vectors into one vector that represents the whole molecule, which is then sent to the prediction head [6]. The readout must give the same result regardless of atom ordering, and it must work for molecules of any size [6], [26].

This work evaluates nine readouts: the three standard non-parametric readouts in PyTorch Geometric [45] global_mean_pool, global_add_pool, and global_max_pool, and the six ordered pairwise concatenations of those aggregations, namely Add+Mean, Mean+Add, Add+Max, Max+Add, Mean+Max, and Max+Mean, each producing a vector of dimension 2 × hidden.

13 Summary

This chapter presented the complete theoretical foundation of the DeepToxiGraph framework. Starting from the pharmaceutical motivation for predicting LD_{50} computationally, four molecular representations were described: SMILES, fingerprints, physicochemical descriptors, and molecular graphs, together with the graph-theoretic matrix formalisms (adjacency, degree, Laplacian, and incidence matrices) on which the GNN architectures depend.

The evolution of computational toxicity prediction was reviewed, from classical QSAR models through multi-task deep neural networks to graph neural networks. Convolutional neural networks were introduced as the conceptual precursor of GNNs, and five GNN architectures were described mathematically: GCN, GraphSAGE, GAT, GIN, and the Graph Transformer. The Transformer attention mechanism and the Graph Transformer were presented, followed by readout functions used in this work. Training

techniques, including batch normalization and dropout, were introduced. Every concept introduced in this chapter is applied directly in the experimental work of Chapter 3.

CHAPTER 3

METHODOLOGY

1 Introduction

This chapter describes the complete experimental methodology behind DeepToxiGraph. Each stage builds on the design decisions established in the previous stage. The chapter begins with an overview of the research framework in Section 2, followed by a description of the dataset in Section 3, a detailed account of each experimental stage in Section 4, and the evaluation metrics used to assess all models in Section 5. The chapter concludes with a summary in Section 6.

2 Research Framework

The DeepToxiGraph framework formulates the prediction of molecular toxicity as a supervised regression problem. Given the SMILES string of a compound as input, the model predicts the acute intraperitoneal toxicity expressed in the standardized unit −log(mol/kg). The framework is structured as six sequential stages, each addressing a specific design question. The answer to each question is fixed as a design decision carried forward into all later stages, ensuring that every architectural choice is supported by experimental evidence.

Stage 1 addresses the following question: among the candidate readout (pooling) strategies, which strategy is the best? A 3-layer GCN with a 128-dimensional hidden layer is used as the base architecture. The readout function selected in Stage 1 is fixed for all experiments.

Stage 2 addresses the following question: among four candidate GNN architectures, which architecture performs best under the same fixed configuration? The readout strategy selected in Stage 1 is applied to all models so that performance differences reflect only architectural design choices.

Stage 3 addresses three successive questions: which depth 4, 5, or 6 message-passing layers gives the best performance for the top two architectures selected from Stage 2? where should batch normalization be inserted within the best-depth model, comparing 10 placement configurations? (c) what is the final fully-regularized training configuration hidden dimension 256, AdamW, Dropout, and gradient clipping for 300 epochs that serves as the GNN baseline carried into the hybrid model of Stage 6?

Stage 4 uses another approach: each molecule is encoded as a 2D image derived from a molecular fingerprint and processed by a CNN. Nine fingerprint types are tested

individually, and the three best-performing types are then fused into a single 3-channel model.

Stage 5 establishes a physicochemical descriptor baseline by training an MLP on 208 RDKit descriptors.

Stage 6 fuses all three molecular representations — graph-based, fingerprint-based, and descriptor-based — into the DeepToxiGraph hybrid model using a sigmoid-gate fusion followed by a shared prediction head.

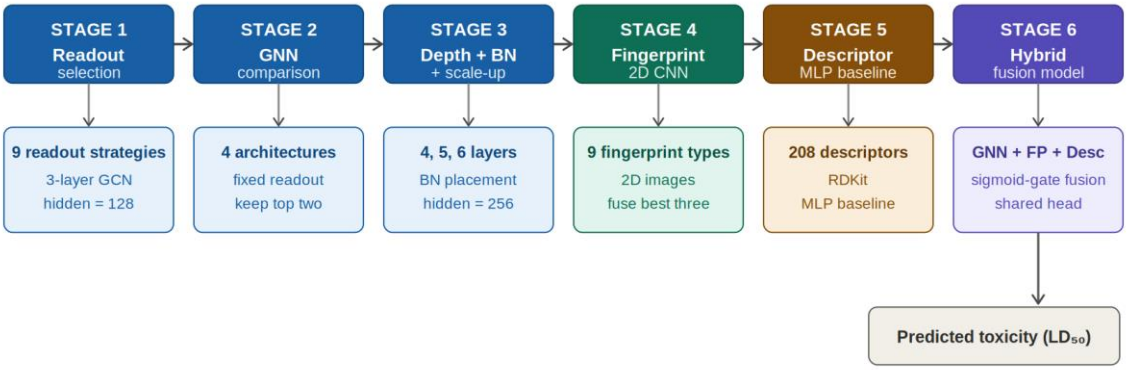


Figure 17: Overview of the six-stage DeepToxiGraph methodology.

3 Dataset and Data Splitting

3.1 The LD_{50} Dataset

All experiments used the dataset of intraperitoneal acute toxicity values compiled from the TOXRIC database [11]. The toxicity endpoint is the 50% lethal dose (LD_{50}), one of the most widely used measures of acute toxicity in the pharmaceutical field.

Table 1: Description of the LD_{50} dataset.

Property	Value
Total compounds	35,299
Training compounds (70%)	24,709
Test compounds (30%)	10,590
Target variable	$-\log(LD_{50})$ [$-\log(mol/kg)$]
Training set mean ± std	3.025 ± 0.696
Test set mean ± std	3.026 ± 0.685
SMILES column	'Canonical SMILES'
Target column	'mouse_intraperitoneal_LD50'

The raw LD_{50} values are expressed in mol/kg and vary considerably across chemical classes. To ensure regression stability, all values are standardised by applying a negative log-10 transformation: −log(mol/kg). A higher −log(mol/kg) value indicates greater toxicity; a lower value indicates lesser toxicity. The log transformation also reduces the influence of outliers during training. The dataset contains 35,299 compounds [11].

Table 2: Sample rows from the LD_{50} dataset sourced from the TOXRIC database.

	TAID	Pubchem CID	IUPAC Name	SMILES	Canonical SMILES	InChIKey	mouse_intraperitoneal_LD50
2	TOX-145	785	benzene-1,4-diol	Oc1ccc(O)cc1	Oc1ccc(O)cc1	QIGBRXMKCJKVMJ-UHFFFAOYS	3,041834651
3	TOX-245	5453	tris(aziridin-1-yl)sulfanylidene-lambda5-phosp	S=P(N1CC1)(N1CC1)N1CC1	S=P(N1CC1)(N1CC1)N1CC1	FOCVUCIESVLUNU-UHFFFAOY	4,235583534
4	TOX-1273	727	1,2,3,4,5,6-hexachlorocyclohexane	ClC1C(Cl)C(Cl)C(Cl)C(Cl)C1Cl	ClC1C(Cl)C(Cl)C(Cl)C(Cl)C1Cl	JLYXXMFPNIAWKQ-UHFFFAOYS	3,366732177
5	TOX-1279	4091	3-(diaminomethylidene)-1,1-dimethylguanidine	CN(C)C(=N)N=C(N)N	CN(C)C(=N)N=C(N)N	XZWYZXLIPXDOLR-UHFFFAOYS	2,641603841
6	TOX-1282	10364	2-methyl-5-propan-2-ylphenol	Cc1ccc(C(C)C)cc1O	Cc1ccc(C(C)C)cc1O	RECUKUPTGUEGMW-UHFFFAO	3,311626674
7	TOX-1289	11092	1-(2-hydroxy-4-methoxyphenyl)ethanone	COc1ccc(C(C)=O)c(O)c1	COc1ccc(C(C)=O)c(O)c1	UILPJVPSNHJFIK-UHFFFAOYSA	2,327917267
8	TOX-1292	219402	2-(3,4-dihydroxy-5-oxooxolan-2-yl)-2-hydroxy	O=CC(O)C1OC(=O)C(O)C1O	O=CC(O)C1OC(=O)C(O)C1O	UYUXSRADSPPKRZ-UHFFFAOY	1,48261524
9	TOX-1293	5281416	6,7-dihydroxychromen-2-one	O=c1ccc2cc(O)c(O)cc2o1	O=c1ccc2cc(O)c(O)cc2o1	ILEDWLMCKZNDJK-UHFFFAOYS	2,074677503
10	TOX-1298	709	3-(4-hydroxy-3-methoxyphenyl)prop-2-enoic a	COc1cc(C=CC(=O)O)ccc1O	COc1cc(C=CC(=O)O)ccc1O	KSEBMYQBYZTDHS-UHFFFAOY	3,000416186
11	TOX-1307	10583	1-ethenyl-1-methyl-2,4-bis(prop-1-en-2-yl)cyc	C=CC1(C)CCC(C(=C)C)CC1C(=C	C=CC1(C)CCC(C(=C)C)CC1C(=C	OPFTUNCRGUEPRZ-UHFFFAOY	2,864785315
12	TOX-1311	896	N-[2-(5-methoxy-1H-indol-3-yl)ethyl]acetamid	COc1ccc2[nH]cc(CCNC(C)=O)c2	COc1ccc2[nH]cc(CCNC(C)=O)c2	DRLFMBDRBRZALE-UHFFFAOY	2,227714728
13	TOX-1318	5281708	7-hydroxy-3-(4-hydroxyphenyl)chromen-4-one	O=c1c(-c2ccc(O)cc2)coc2cc(O)cc	O=c1c(-c2ccc(O)cc2)coc2cc(O)cc	ZQSIJRDFPHDXIC-UHFFFAOYS	2,104215592
14	TOX-1326	2240	1,5,9-trimethyl-11,14,15,16-tetraoxatetracyclo	CC1CCC2C(C)C(=O)OC3OC4(C)(	CC1CCC2C(C)C(=O)OC3OC4(C)(	BLUAFEHZUWYNDE-UHFFFAOY	2,258198804
15	TOX-1337	3473	5-hydroxy-1-(4-hydroxy-3-methoxyphenyl)dec	CCCCCC(O)CC(=O)CCc1ccc(O)c	CCCCCC(O)CC(=O)CCc1ccc(O)c	NLDDIKRKFXEWBK-UHFFFAOYS	3,704748396
16	TOX-1342	5538	3,7-dimethyl-9-(2,6,6-trimethylcyclohexen-1-y	CC(C=CC1=C(C)CCCC1(C)C)=C	CC(C=CC1=C(C)CCCC1(C)C)=C	SHGAZHPCJJPHSC-UHFFFAOY	3,278486808
17	TOX-1343	471	2-(3,4-dihydroxyphenyl)-3,5,7-trihydroxy-2,3-	O=C1c2c(O)cc(O)cc2OC(c2ccc(O	O=C1c2c(O)cc(O)cc2OC(c2ccc(O	CXQWRCVTCMQVQX-UHFFFAOY	2,489800066
18	TOX-1345	2198	(6-hydroxy-8-methyl-8-azabicyclo[3.2.1]octan	CN1C2CC(OC(=O)C(CO)c3ccccc	CN1C2CC(OC(=O)C(CO)c3ccccc	WTQYWNWRJNXDEG-UHFFFAC	2,940764013
19	TOX-1346	2548	N-[(4-hydroxy-3-methoxyphenyl)methyl]-8-me	COc1cc(CNC(=O)CCCCC=CC(C)	COc1cc(CNC(=O)CCCCC=CC(C)	YKPUWZUDDOIDPM-UHFFFAOY	4,379467946
20	TOX-1358	159287	2-(4-hydroxy-3,5-dimethoxyphenyl)chromeny	COc1cc(-c2[o+]c3cc(O)cc(O)c3cc	COc1cc(-c2[o+]c3cc(O)cc(O)c3cc	KZMACGJDUUWFCH-UHFFFAO	1,906379614
21	TOX-1366	3576890	9,10,15,18-tetrahydroxy-12,12-dimethyl-6-me	C=C1C(=O)C23C(O)C1CCC2C12(	C=C1C(=O)C23C(O)C1CCC2C12(	SDHTXBWLVGWJFT-UHFFFAOY	4,01755561
22	TOX-1371	124327	2-heptadec-10-enyl-6-methoxycyclohexa-2,5-	CCCCCCC=CCCCCCCCCC1=C	CCCCCCC=CCCCCCCCCC1=C	YYCCUFKHCNSRIA-UHFFFAOY	4,168693477
23	TOX-1375	2735	3-[2-[7a-methyl-1-(6-methylheptan-2-yl)-2,3,3a	C=C1CCC(O)CC1=CC=C1CCCC(	C=C1CCC(O)CC1=CC=C1CCCC(	QYSXJUFSXHHAJI-UHFFFAOYS	3,45152457
24	TOX-1379	220520	10,11-dimethoxy-4a,5,5a,7,8,13a,15,15a,15b,	COc1cc2c(cc1OC)C13CCN4CC5(	COc1cc2c(cc1OC)C13CCN4CC5(	RRKTZKIUPZVBMF-UHFFFAOYS	3,803623392
25	TOX-1385	634470	3,4,5,19-tetramethoxy-9,10-dimethyl-15,17-dio	COc1cc2c(c(OC)c1OC)-c1c(cc3c(	COc1cc2c(c(OC)c1OC)-c1c(cc3c(	ZWRRJEICIPUPHZ-UHFFFAOYS	3,028519116
26	TOX-1396	135444779	2-[[4-[(2-amino-4-oxo-3H-pteridin-6-yl)methyla	Nc1nc(=O)c2nc(CNc3ccc(C(=O)N	Nc1nc(=O)c2nc(CNc3ccc(C(=O)N	OVBPIULPVIDEAO-UHFFFAOYS	3,715417339
27	TOX-1410	425990	[6-hydroxy-8-methyl-3-[3,4,5-trihydroxy-6-(hy	CC12CC3(O)OC(O1)C1(COC(=O)	CC12CC3(O)OC(O1)C1(COC(=O)	YKRGDOXKVOZESV-UHFFFAOY	2,133887955
28	TOX-1443	360849	22,33-dimethoxy-13,28-dimethyl-2,5,7,20-tetra	COc1ccc2cc1Oc1ccc(cc1)CC1c3(	COc1ccc2cc1Oc1ccc(cc1)CC1c3(	YVPXVXANRNDGTA-UHFFFAOY	3,686077583
29	TOX-1445	75055024	[2-(3,4-dihydroxyphenyl)-6-(hydroxymethyl)-5	CC1OC(OC2C(OC(=O)C=Cc3ccc(	CC1OC(OC2C(OC(=O)C=Cc3ccc(	FPOYEEKZOOLVJA-UHFFFAOYS	2,09522237
30	TOX-1470	12912318	2-[2-[[3,12-dihydroxy-17-(2-hydroxy-6-methyl	CC(C)=CCCC(C)(O)C1CCC2(C)C	CC(C)=CCCC(C)(O)C1CCC2(C)C	AGBCLJAHARWNLA-UHFFFAOY	2,767778689
31	TOX-1471	12901617	2-[4,5-dihydroxy-2-[[12-hydroxy-17-(2-hydroxy	CC(C)=CCCC(C)(O)C1CCC2(C)C	CC(C)=CCCC(C)(O)C1CCC2(C)C	RWXIFXNRCLMQCD-UHFFFAOY	2,797973475
32	TOX-1477	432116	2-[[3,12-dihydroxy-4,4,8,10,14-pentamethyl-1	CC(C)=CCCC(C)(OC1OC(CO)C(C	CC(C)=CCCC(C)(OC1OC(CO)C(C	YURJSTAIMNSZAE-UHFFFAOYS	3,296190505
33	TOX-1478	4483443	2-[2-[[3,12-dihydroxy-17-(2-hydroxy-6-methyl	CC(C)=CCCC(C)(O)C1CCC2(C)C	CC(C)=CCCC(C)(O)C1CCC2(C)C	UZIOUZHBUYLDHW-UHFFFAOY	2,77654073
34	TOX-1486	262500	2-[5-hydroxy-6-(hydroxymethyl)-2-[(10,14,16,	CC1CCC2C(C)C3C(CC4C5CC=C	CC1CCC2C(C)C3C(CC4C5CC=C	ZGVSETXHNHBTRK-UHFFFAOYS	4,394038832
35	TOX-1495	4752	28-(2,3-dihydroxy-2-methylpropyl)-18-hydroxy	CC1NC(=O)C(CC(C)(O)CO)NC(=(	CC1NC(=O)C(CC(C)(O)CO)NC(=(	KPKZJLCSROULON-UHFFFAOY	5,595981501
36	TOX-1496	108174	15-hydroxy-7-[6-(hydroxymethyl)-3-(3-methyl	C=C1C2CCC3C4(C)CC(OC5OC((	C=C1C2CCC3C4(C)CC(OC5OC((	AQFATIOBERWBDY-UHFFFAOY	4,861649926
37	TOX-1498	2012	[8-acetyloxy-11-ethyl-5,7,14-trihydroxy-6,16,	CCN1CC2(COC)C(O)CC(OC)C34	CCN1CC2(COC)C(O)CC(OC)C34	XFSBVAOIAHNAPC-UHFFFAOYS	6,283834599
38	TOX-1523	4437077	7,7,12,16-tetramethyl-6-(methylamino)-15-[1-(	CNC(C)C1C(O)CC2(C)C3CCC4C	CNC(C)C1C(O)CC2(C)C3CCC4C	GMNAPBAUIVITMI-UHFFFAOYSA	4,641158212
39	TOX-1525	108130	3,4,5,19-tetramethoxy-9,10-dimethyl-15,17-dio	COc1cc2c(c(OC)c1OC)-c1c(cc3c(	COc1cc2c(c(OC)c1OC)-c1c(cc3c(	RTZKSTLPRTWFEV-UHFFFAOYS	2,301541077
40	TOX-1536	4581422	[7-(2-methylbut-2-enoyloxy)-5,6,7,8-tetrahydr	CC=C(C)C(=O)OC1CCN2CC=C(C(	CC=C(C)C(=O)OC1CCN2CC=C(C(	MVWPTZQHBOWRTF-UHFFFAOY	3,104337996

This table shows the dataset that was used in the experiments. Each row in the table represents one chemical compound. Each column represents one attribute of that compound. The attributes include TAID (a unique identifier for each compound), PubChem ID (another unique identifier), chemical name (the IUPAC name of the compound), SMILES (a string representation of the compound's molecular structure), canonical SMILES (the same as SMILES but in a canonical form), InChIKey (another

unique identifier for the compound), and mouse_intraperitoneal_LD50 (represents how toxic the compound is).

3.2 Molecular Graph Construction

For each GNN experiment, each SMILES string is converted to a PyTorch Geometric *Data* object through a graph construction algorithm based on the RDKit library.

Each atom is represented by a 14-dimensional feature vector encoding atomic number, degree, formal charge, aromaticity, ring membership, total hydrogen count and hybridization state, Gasteiger partial charge, and membership in rings of size 3, 4, 5, and 6.

Each bond is represented by a 7-dimensional feature vector encoding bond type (single, double, triple, aromatic), conjugation, ring membership, and stereo configuration. All features are normalized or one-hot encoded and any NaN or infinite values are replaced with zero.

3.3 Train and Test Split

The dataset is split into a training set (70%, containing 24,709 compounds) and a test set (30%, containing 10,590 compounds) using a random split with a fixed seed of 42. This split is applied to all experiments, ensuring that all models are evaluated on the same held-out test set.

4 Model Development

The experiments are arranged from the simplest design question to the most complex. Each subsection describes the experimental setup and procedure for one stage of the methodology.

4.1 Stage 1 Finding the Best Readout Function

For this evaluation, a fixed base architecture was used: a 3-layer Graph Convolutional Network (GCNConv) with a hidden dimension of 128, an Adam optimiser with a learning rate of 0.001, MSE loss, a ReduceLROnPlateau scheduler (factor = 0.5, patience = 20), a batch size of 64, and 200 training epochs. After the three convolutional layers the readout function is applied to produce a single graph-level vector, which is then

passed through an MLP head of sizes readout_dim -> 256 -> 64 -> 1. All hyperparameters except the readout function are held constant, so any performance differences can be attributed solely to the pooling strategy.

Nine readout strategies were evaluated, ranging from simple global aggregations to all ordered pairwise concatenations:

Table 3: The nine readout strategies evaluated in Stage 1.

	Readout Strategy	**Description**
1	Global Add Pool	global_add_pool sums all node feature vectors into one graph-level vector
2	Global Mean Pool	global_mean_pool averages all node feature vectors into one graph-level vector
3	Global Max Pool	global_max_pool takes the element-wise maximum across all node features
4	Add+ Mean concatenation	cat (global_add_pool, global_mean_pool) → 2×hidden-dimensional vector
5	Mean+Add concatenation	cat (global_mean_pool, global_add_pool) → 2×hidden-dimensional vector
6	Add + Max concatenation	cat (global_add_pool, global_max_pool) → 2×hidden-dimensional vector
7	Max + Add concatenation	cat (global_max_pool, global_add_pool) → 2×hidden-dimensional vector; selected readout
8	Mean+ Max concatenation	cat(global_mean_pool,global_max_pool) → 2×hidden-dimensional vector
9	Max+ Mean concatenation	cat(global_max_pool,global_mean_pool) → 2×hidden-dimensional vector

The (max+add) concatenation readout achieved the best test performance and is therefore fixed as the readout function for all GNN experiments.

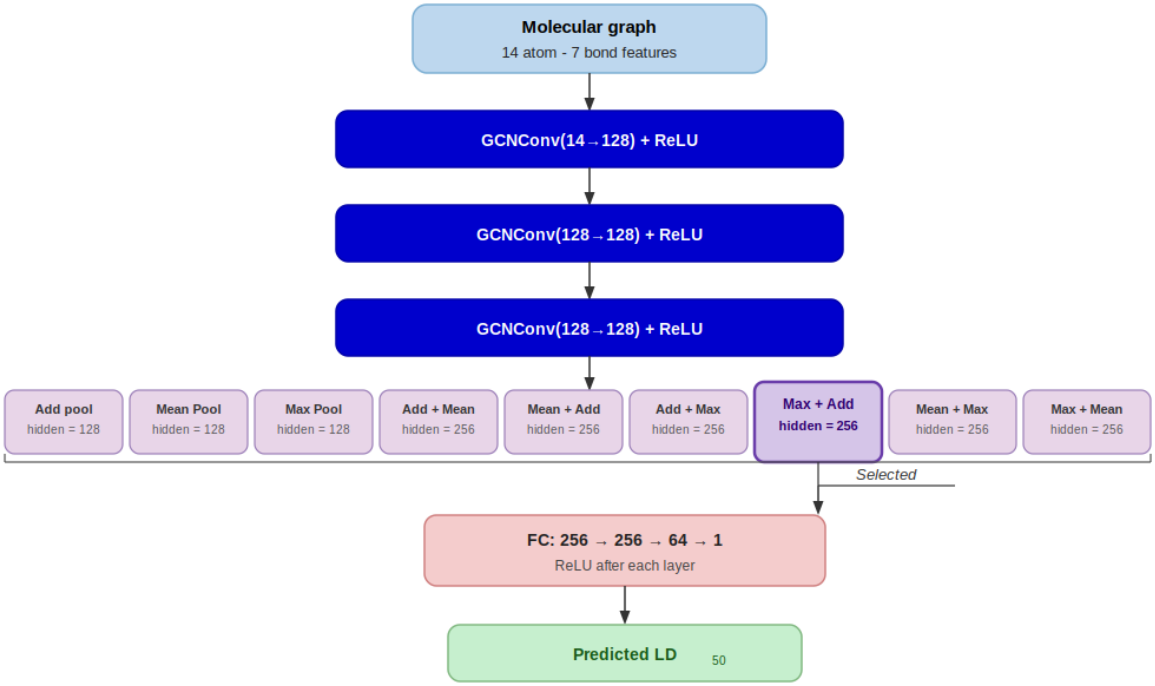


Figure 18: Nine readout strategies.

4.2 Stage 2 Comparing GNN Architectures

With the readout function fixed to the max+add concatenation, the second stage benchmarks four GNN architectures on the same dataset split. All models use the following identical configuration: three message-passing layers, 128-dimensional hidden representations, max+add concatenation readout, no batch normalization, Adam optimiser (lr = 0.001), ReduceLROnPlateau scheduler (factor = 0.5, patience = 10), batch size of 64, and 200 training epochs. Any performance differences between models are because of their architectural design alone.

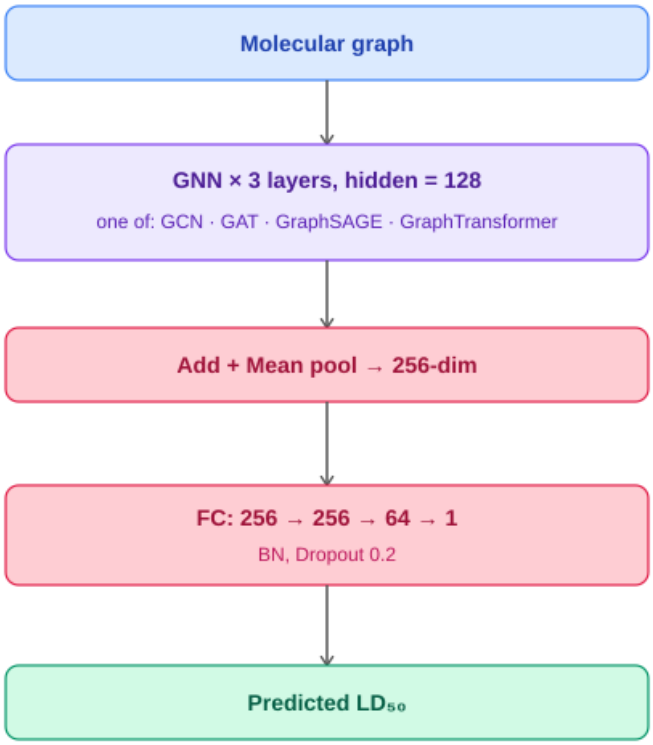


Figure 19: Stage 2: four GNN architectures.

All four architectures use a shared MLP head of sizes 2×hidden (256) → 256 → 64 → 1 applied after the readout. After evaluation, the two best-performing architectures Graph Isomorphism Network and Graph Transformer are selected and carried forward to Stage 3 for further investigation.

4.3 Stage 3 Depth Search, Batch Normalization Placement and Scale-Up

Stage 3 is conducted in three successive sub-experiments, each building on the result of the previous one.

4.3.1 Depth Search

The two architectures selected from Stage 2 Graph Isomorphism Network and Graph Transformer are re-evaluated at depths of 4, 5, and 6 message-passing layers. All other hyperparameters remain identical to Stage 2: hidden dimension 128, max+add readout, no batch normalisation, Adam (lr = 0.001), ReduceLROnPlateau (factor = 0.5, patience = 10), batch size 64, and 200 training epochs. Six runs in total: GIN-L4, GIN-L5, GIN-L6, GT-L4, GT-L5, and GT-L6. The best-performing depth configuration is carried forward to the batch normalization placement study.

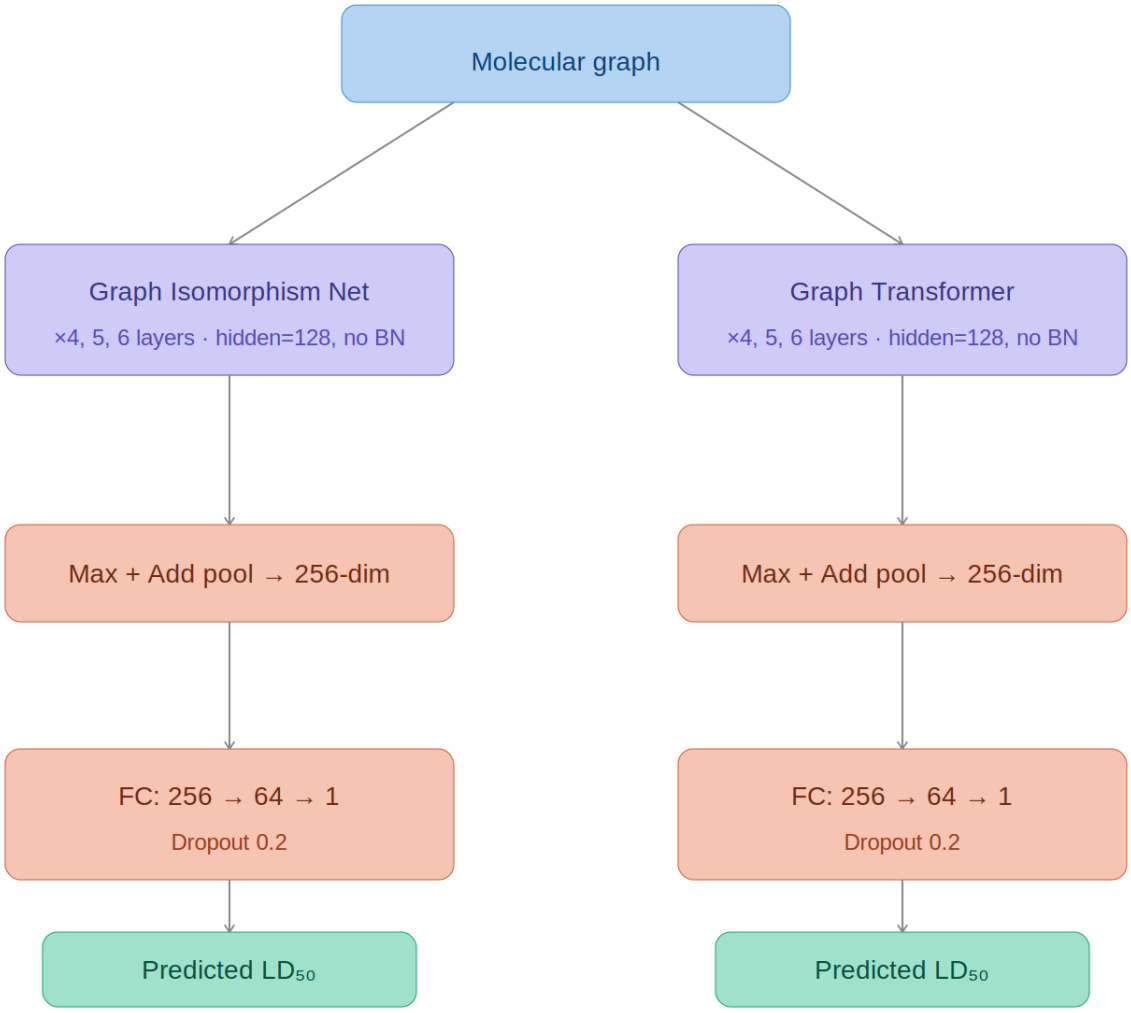


Figure 20: Stage 3a: depth search for Graph Isomorphism Network and Graph Transformer at 4, 5, and 6 layers.

Graph Transformer at four layers is selected as the best-performing configuration, and it is used for the study on batch normalization placement.

4.3.2 Batch Normalization Placement Study

A study of batch normalization (BN) placement is conducted on the Graph Transformer-4L architecture. The forward pass follows the Conv → BN → ReLU and The FC head applies the pattern Linear → BN → ReLU.

Eleven configurations are evaluated:

Table 4: Batch normalization placement configurations for Graph Transformer-4L.

Configuration	BN after conv layers	BN after FCs	Notes
GT4L_BN_FC1_only	None	FC1 only	
GT4L _BN_FC2_only	None	FC2 only	
GT4L _BN_FC_both	None	FC1 + FC2	
GT4L _BN_conv1	Layer 1 only	None	
GT4L _BN_conv2	Layer 2 only	None	
GT4L _BN_conv3	Layer 3 only	None	
GT4L _BN_conv4	Layer 4 only	None	
GT4L_BN_convs_all	All 4 layers	None	
GT4L_BN_all	All 4 layers	FC1 + FC2	Selected configuration
GT4L_noBN (baseline)	None	None	

All other hyperparameters are unchanged from the depth search. The configuration GT4L_BN_all which applies batch normalization after every convolutional layer and after both fully-connected layers in the head is selected as the standard training configuration for all GNN experiments.



Figure 21: Ten batch normalization placement configurations evaluated on Graph Transformer-4L.

4.3.3 Final GNN Configuration

A selected architecture Graph Transformer is trained with a hidden dimension of 256 for 300 epochs, with the batch normalization strategy fixed to BN_all. Additional regularization is applied throughout: AdamW optimiser (weight decay = 1×10^{-4}), Dropout ($p = 0.2$) in the FC head, and gradient norm clipping at 1.0. A ReduceLROnPlateau scheduler (factor = 0.5, patience = 15) is used to reduce the learning rate when the test R^2 stops improving. The random seed is reset identically before each run. This configuration

defines the final, fully regularized standalone GNN models that are used as the GNN branch encoders in the hybrid model.

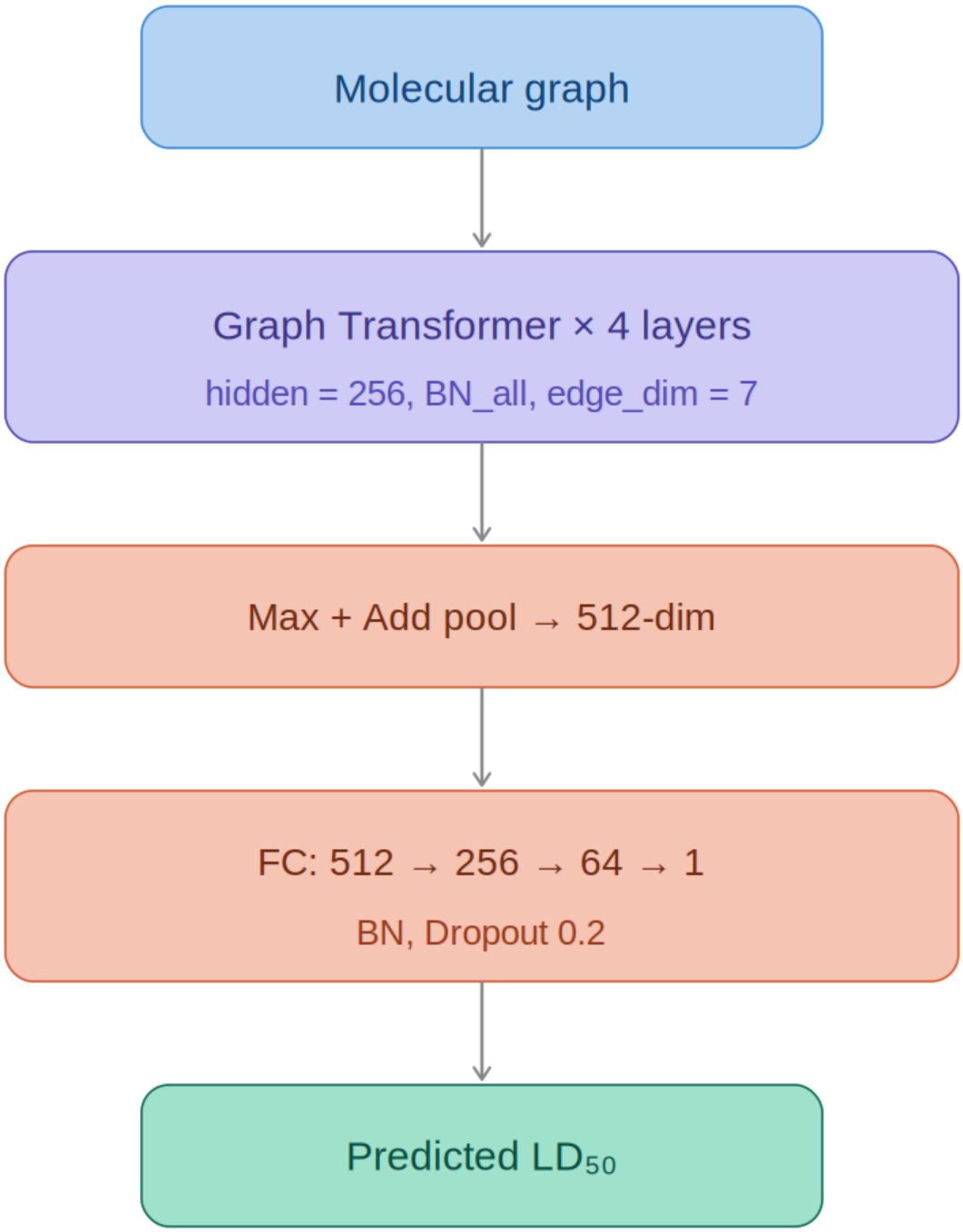


Figure 22: GraphTransformer_4L architecture.

4.4 Stage 4 Fingerprint Based 2D CNN Models

In addition to the graph-based experiments, a second approach uses molecular fingerprints encoded as 2D images and processed by a convolutional neural network. This model is developed both as a standalone baseline and as a source of complementary features for the hybrid model in Stage 6.

4.4.1 Fingerprint Representation

For each of nine fingerprint types (ECFP4, ECFC4, FCFP4, FCFC4, RDK5, RDK6, RDK7, MACCS, Avalon), a 1,024-bit binary vector is computed using RDKit with NBITS = 1,024. This vector is reshaped into a 32×32 matrix, representing a single-channel 2D image. This representation allows the CNN to capture spatial relationships between fingerprint bits that encode structural features of the molecule.

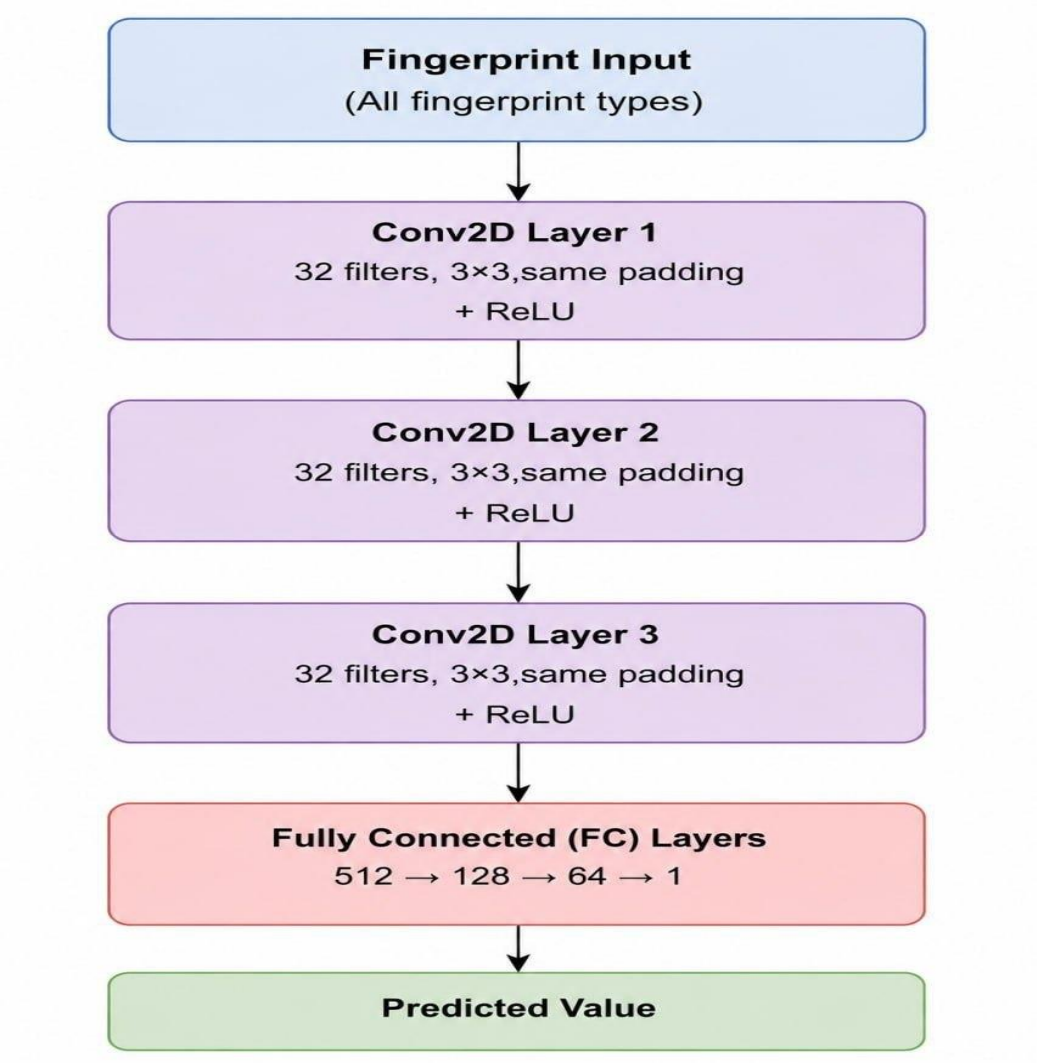


Figure 23: CNN architecture used for all fingerprints.

4.4.2 Individual Fingerprint Evaluation

Each of the nine fingerprint types is trained independently using the CNN architecture described before. The nine types evaluated are: ECFP4, ECFC4, FCFP4, FCFC4, RDK5, RDK6, RDK7, MACCS, and Avalon.

4.4.3 Fingerprint Fusion: Concatenated 3-Channel 2D CNN

The three best-performing fingerprint types, Avalon, RDK5, and ECFC4, are combined to create a single multi-channel input. Each fingerprint type represents the molecule from a different structural perspective.

The concatenated model uses the same architecture as the individual models except that the first convolutional layer takes three channels as input: Conv2d (3→32). All remaining layers, the training procedure, and the evaluation protocol are identical to the individual fingerprint experiments.

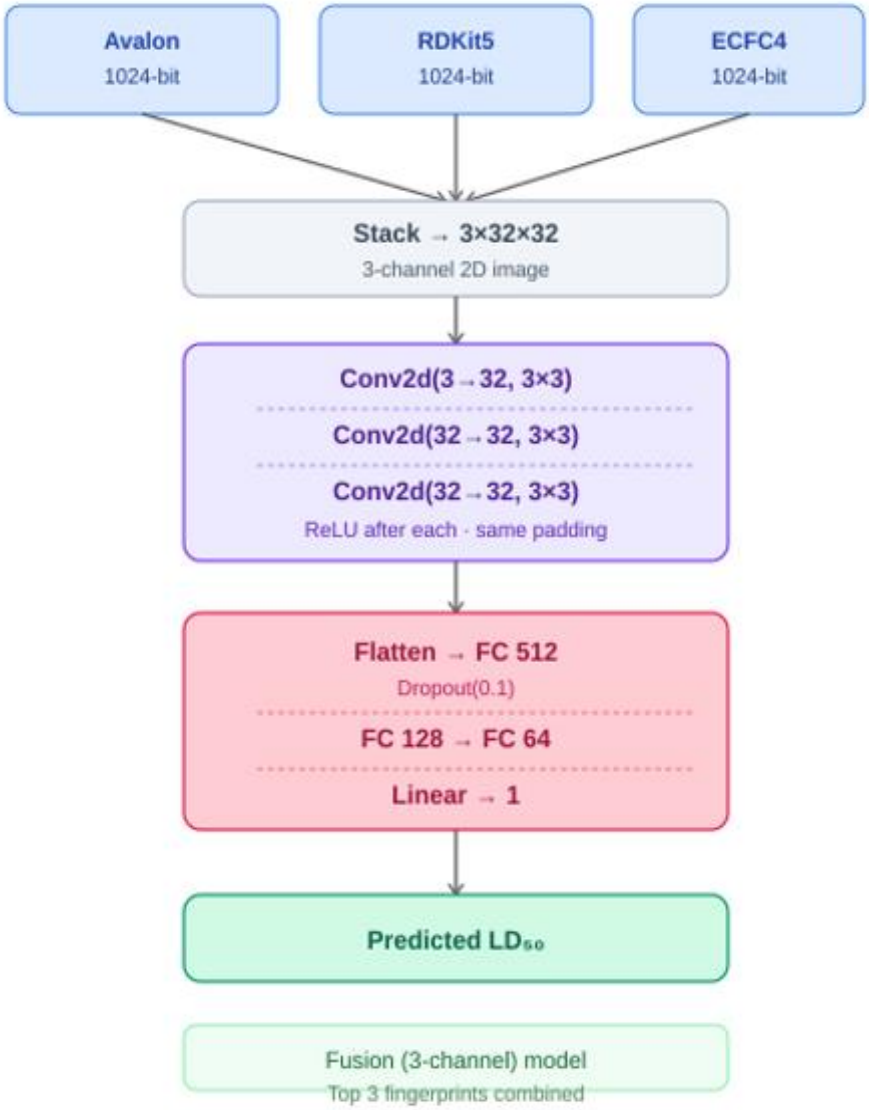


Figure 24: Fusion 3-channel model combining Avalon, RDKit5, and ECFC4 fingerprints.

4.5 Stage 5 Physicochemical Descriptor MLP Baseline

A third baseline model is constructed using only the physicochemical descriptors of each molecule, providing a representation complementary to both the graph-based and fingerprint-based approaches.

4.5.1 Descriptor Computation

All 208 RDKit descriptors [28] are calculated for each molecule using the *Descriptors* module. Any descriptors with NaN or infinite values are replaced with zero.

The model is trained with the Adam optimizer (initial lr = 1×10^{-3}, weight decay = (1×10^{-4}). Training runs for a maximum of 500 epochs with batch size 64.

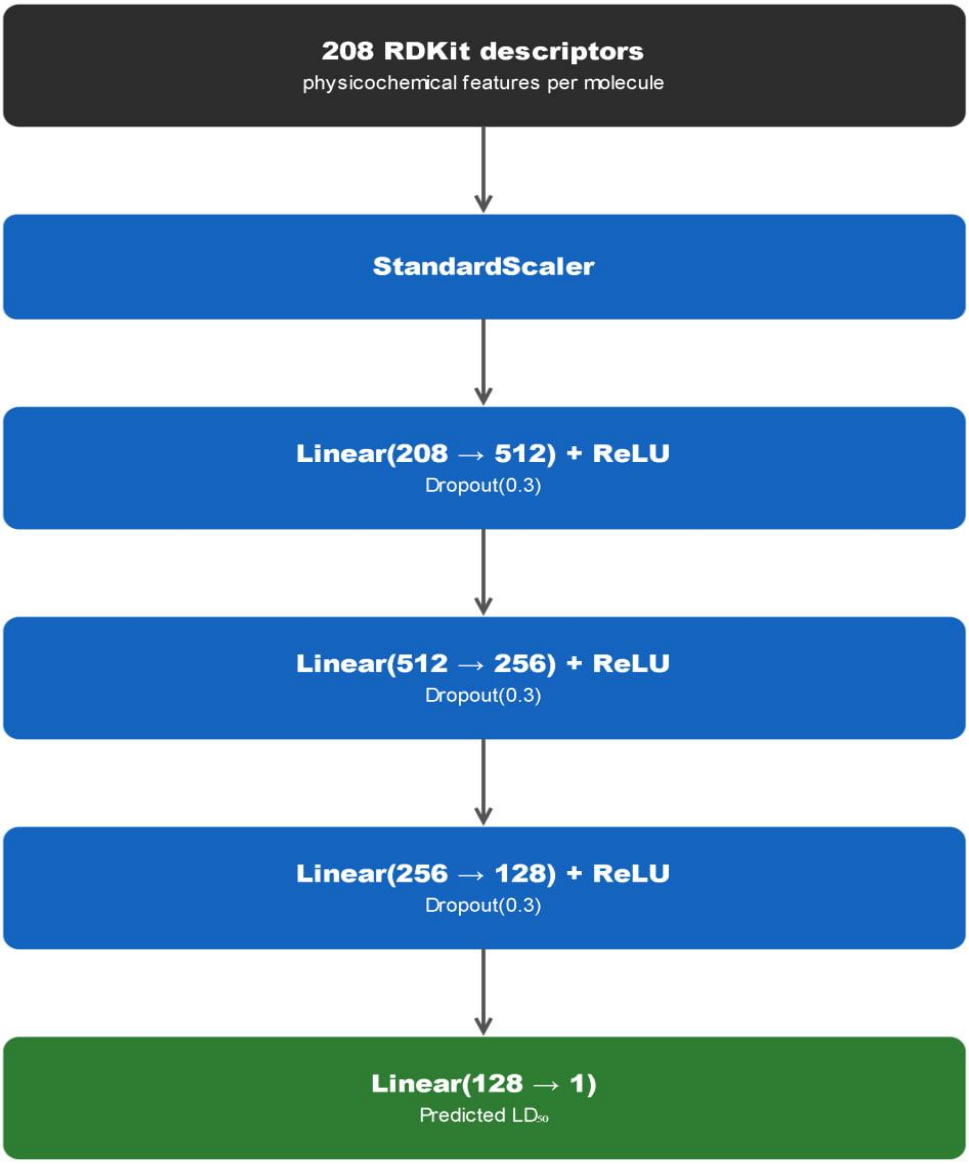


Figure 25: Descriptor MLP architecture.

4.6 Stage 6 DeepToxiGraph Hybrid Model

Stage 6 develops the DeepToxiGraph hybrid model, which fuses three molecular representations graph-based, fingerprint-based, and descriptor-based into a single unified prediction. The motivation for fusion is that each representation encodes different aspects of molecular structure that are not fully captured by any single view alone. Different

molecular representations encode different aspects of molecular structure, and a model with access to all three views is expected to generalize more robustly.

4.6.1 Architecture Overview

The DeepToxiGraph hybrid model consists of three encoding branches and a shared prediction head. Each branch independently encodes its molecular representation into a 256-dimensional embedding vector. The three embeddings are concatenated into a single 768-dimensional vector (3 × 256), which is then passed through a sigmoid gate and a three-layer FC head to produce the final log LD_{50} prediction.

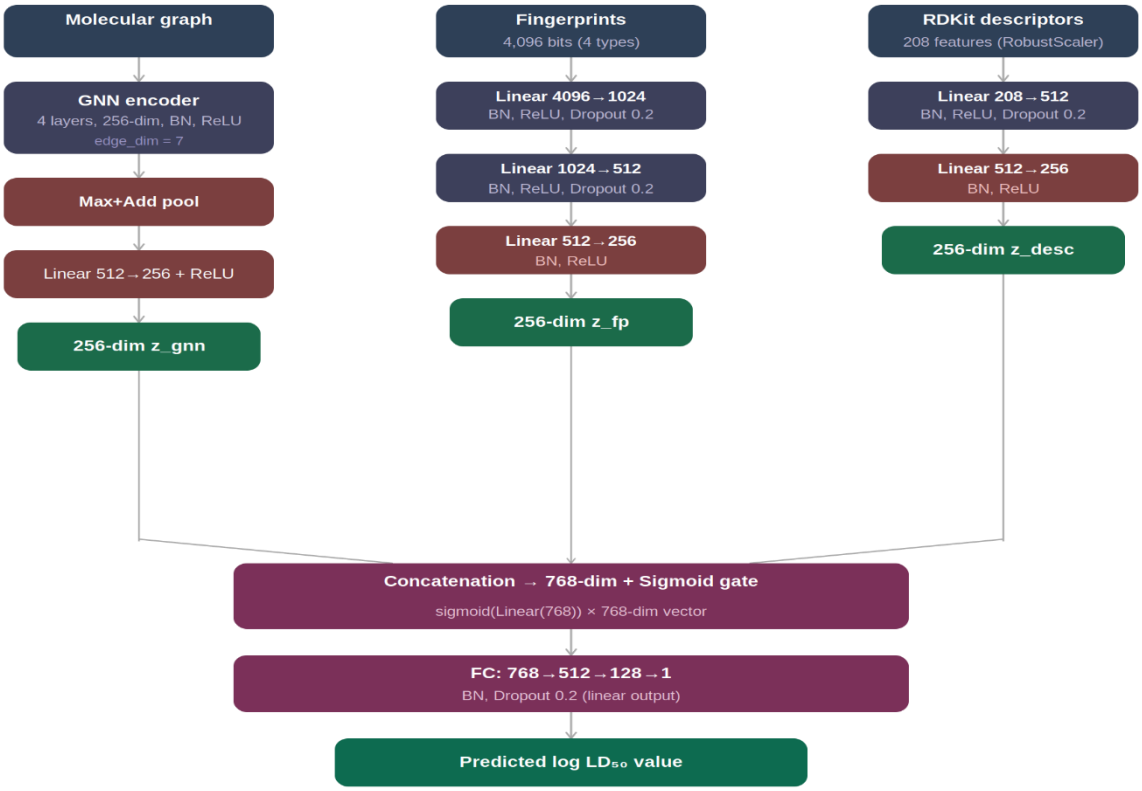


Figure 26: DeepToxiGraph hybrid model.

4.6.2 GNN Branch

The GNN branch uses the same architecture from: four message-passing layers with a 256-dimensional hidden representation, batch normalization after every convolutional layer, and ReLU activations. The Max + Add concatenation is used. A single linear projection layer (512 → 256) followed by ReLU maps this vector to the 256-

dimensional branch embedding used in the fusion step. The model uses a Graph Transformer encoder.

4.6.3 Fingerprint MLP Branch

The fingerprint branch concatenates three different types of binary fingerprints into a single vector of 3,072 bits. The three types of fingerprints are: Avalon (1,024 bits), RDKit (1,024 bits), and ECFP4 Morgan fingerprints with a radius of 2 (1,024 bits). A fourth type of fingerprint, Atom-Pair fingerprints (1,024 bits), is also added to the concatenation to produce a 4,096-bit vector. Atom-Pair fingerprints encode the distances between every pair of atoms within the molecule and of what type each of those atoms is, indicating the larger shape of the molecule - information that is not encoded by the circular representations of the other three fingerprint types. This 4,096-bit vector is passed through a four-layer MLP with dimensions 4096 → 1024 → 512 → 256, with batch normalization, ReLU activation functions, and dropout with a probability of p = 0.2. The output of this MLP is the 256-dimensional embedding.

4.6.4 Descriptor MLP Branch

The descriptor branch processes the 208 RDKit physicochemical descriptors, standardized using a RobustScaler fitted on the training set. The branch uses a two-layer MLP:

208 → 512 → 256, with batch normalization, ReLU activations, and Dropout (p = 0.2) . The output is a 256-dimensional embedding.

4.6.5 Concatenation Fusion and Prediction Head

The three 256-dimensional branch embeddings z_gnn, z_fp, and z_desc are concatenated into a 768-dimensional vector. This vector is passed through a sigmoid gate a learned linear layer that produces a 768-dimensional weight vector, which is multiplied element-wise with the concatenated embedding to emphasise the most informative dimensions before being passed into the prediction head. The prediction head consists of three fully-connected layers: 768 → 512 → 128 → 1, with batch normalization and Dropout (p = 0.2) and a linear output for the final layer. The scalar output is the predicted log LD_{50} value of the molecule.

4.6.6 Training Configuration

The model is trained using AdamW with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-4} for 300 epochs using batches of 64 examples. The learning rate starts at 1×10^{-4} and increases over the first 15 epochs to the target learning rate. After reaching the target learning rate, it is halved if the model's loss does not decrease for 25 consecutive epochs, with a minimum learning rate of 5×10^{-6}. The clipping value for gradient clipping is 1.0 and the dropout value is 0.2 for all fully connected layers. All experiments use the same random seed of 42.

5 Evaluation Metrics

All models are evaluated on the same test set using three complementary metrics to determine their performance: Root Mean Squared Error (RMSE) [46], the coefficient of determination (R^2) [47], and the Pearson correlation coefficient (r) [48]. Using three different metrics allows for a more complete understanding of the model's performance than using only a single metric.

Root Mean Squared Error (RMSE)

The Root Mean Squared Error takes the square root of the average of the squared differences between predicted and true values. The result is expressed in the same physical units as the targeted variable - directly interpretable with a heavy penalty placed on large prediction errors [46]. Lower values indicate better performance as a predictor. The RMSE is defined by:

$$RMSE = \sqrt{((1/n) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2)} \quad (3.1)$$

where n is the number of compounds in the test set, y_i is the experimental LD_{50} value for compound i, and $\hat{y}_i$ is the model's prediction for the same compound.

Coefficient of Determination (R^2)

The coefficient of determination measures the proportion of variance in the target variable that is explained by the model [47]. It ranges from $-\infty$ to 1; a value of 1 indicates perfect prediction, a value of 0 indicates that the model performs no better than predicting the mean of the training set, and a negative value indicates a model worse than the mean predictor. R^2 is computed by:

$$R^2 = 1 - (\sum_i (y_i - \hat{y}_i)^2) / (\sum_i (y_i - \bar{y})^2) \qquad (3.2)$$

where $\bar{y}$ is the mean of the experimental values in the test set. R^2 is selected as the primary criterion in this work for choosing the final model and is reported alongside RMSE for every experiment.

Pearson Correlation Coefficient (r)

The Pearson correlation coefficient measures the degree to which the predictions follow the same trend as the real values [48]. It takes a value between -1 and 1, with 1 indicating a perfect agreement between the two sets of values, 0 indicating no link between the two variables, and -1 indicating that the two variables have an opposite trend to each other. The coefficient is useful in determining whether the model correctly ranks the molecules from least to most toxic. It is calculated by:

$$r = (\sum_i (y_i - \bar{y})(\hat{y}_i - \hat{\bar{y}})) / \sqrt{(\sum_i (y_i - \bar{y})^2 \cdot \sum_i (\hat{y}_i - \hat{\bar{y}})^2)} \qquad (3.3)$$

where $\bar{y}$ and $\hat{\bar{y}}$ are respectively the means of the experimental and predicted values in the test set.

Each model is evaluated on the test set. The model that achieved the best R^2 score on the test set is saved and used for the final evaluation

6 Summary

This chapter describes the complete experimental methodology and dataset splitting of DeepToxiGraph as a six-stage process, each stage addressing one question.

In Stage 1, nine different readout strategies are evaluated with the same fixed GNN model to determine which strategy would be used in the remaining experiments. The max+add concatenation strategy is chosen for the remaining experiments.

In Stage 2, four different GNN models are evaluated with the same fixed conditions to determine which two would be used in the remaining experiments. The Graph Isomorphism Network and Graph Transformer models are chosen as the two best performing models.

In Stage 3, different values for the depth of the models, the placement of batch normalization layers within the models, and the training configuration of the two chosen models are evaluated. Depths of 4, 5, and 6 layers are evaluated, with 10 different

placements of batch normalization within the Graph Transformer models of depth 4. Then the model is trained with a hidden dimension of 256 for 300 epochs using AdamW, dropout, and gradient clipping.

In Stage 4, nine different fingerprint types are evaluated for encoding the molecules as 2D images that are passed through a shared CNN. The three best-performing fingerprint types are fused into a new model that also incorporates a shared CNN to evaluate whether the fusion of these three fingerprints improves performance.

In Stage 5, an MLP is trained on 208 physicochemical descriptors to establish a baseline for **toxicity** predictions.

In Stage 6, the graph, fingerprint, and descriptor representations from Stages 3, 4, and 5 are fused together into a single DeepToxiGraph model. The embeddings for the graphs, the fingerprints, and the descriptors are each encoded into 256-dimensional vectors, and these three vectors are concatenated into a single 768-dimensional vector, passed through a sigmoid gate, and then through a three-layer fully-connected head to make a single prediction.

CHAPTER 4

RESULTS AND DISCUSSIONS

1 Introduction

In this chapter, we present the implementation and results of the experimental stages described in Chapter 3. Each stage builds on the design decisions of the previous one, progressing from a readout strategy to standalone GNN models, fingerprint-based 2D CNN models, a physicochemical descriptor MLP baseline, and finally the hybrid model. The dataset, molecular graph construction, and evaluation metrics used throughout are described in detail in Chapter 3 (Sections 3.2 and 3.4). Results are reported in terms of R^2, RMSE, and Pearson correlation coefficient (r) on the test set.

2 Stage 1 Readout Strategy Results

As described in Section 3.3.1, nine readout strategies are evaluated on a fixed 3-layer GCN with 128 hidden units and no batch normalization. Figure 27 presents the resulting R^2 scores on the test set for each of the readout strategies. The nine different readout strategies evaluated were: global mean pooling, global add pooling, global max pooling, and all six ordered pairwise concatenations of those aggregations: Add+Mean, Mean+Add, Add+Max, Max+Add, Mean+Max, and Max+Mean, each producing a vector of dimension 2×hidden = 256.

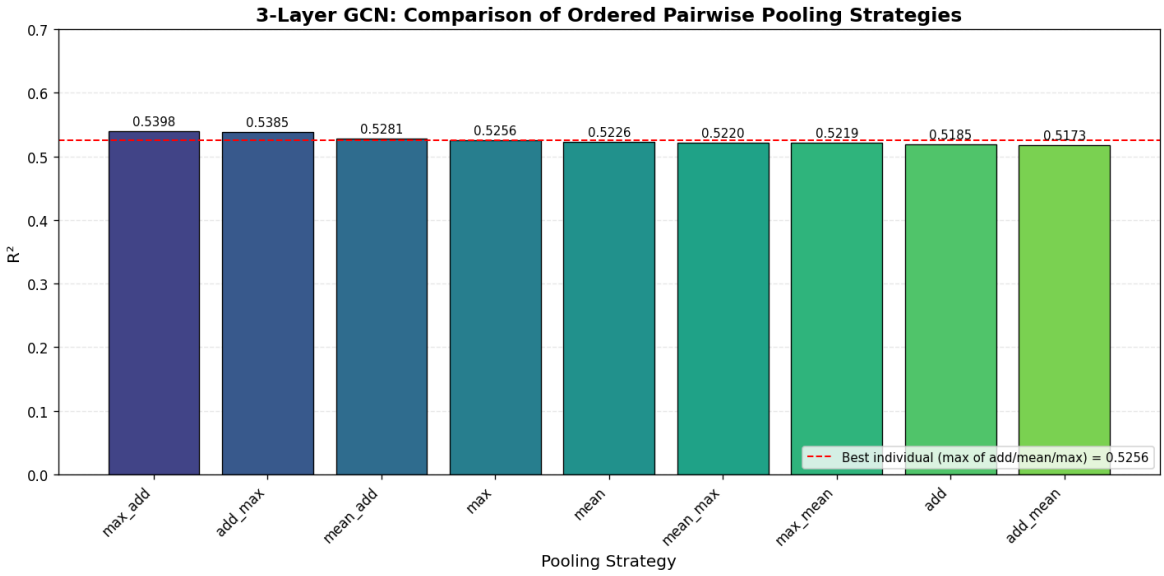


Figure 27: Test R^2 comparison of nine readout strategies on a 3-layer GCN.

As presented in Figure 27, the combined **Max + Add** readout achieved the highest value of **R^2 at 0.5398**, which is higher than the values obtained by the remaining readouts:

Add+Max has an R^2 of 0.5385, Mean+Add has an R^2 of 0.5281, global max pool has an R^2 of 0.5256, global mean pool has an R^2 of 0.5226, Mean+Max has an R^2 of 0.5220, Max+Mean has an R^2 of 0.5219, global add pool has an R^2 of 0.5185, and Add+Mean has an R^2 of 0.5173. These findings are in agreement with the literature, where it is stated that pooling processes combining the max and add pooling methods work best, as the max pooling helps to capture the most prominent atomic features in the molecule, while the add pooling helps to determine the size of the molecule, and combining these two methods allows for a more thorough representation of the molecule.

Table 5: Test R^2 scores for all readout strategies.

Readout Strategy	R^2
Max + Add (combined)	**0.5398**
Add + Max	0.5385
Mean + Add	0.5281
Global Max Pool	0.5256
Global Mean Pool	0.5226
Mean +Max	0.5220
Max + Mean	0.5219
Global Add Pool	0.5185
Add + Mean	0.517

3 Stage 2 GNN Architecture Results

Using the Max + Add readout fixed in Stage 1, four GNN architectures were benchmarked under the same conditions (Section 4.2): **GraphSAGE, GIN**, **GATv2**, and **GraphTransformer.** Figure 27 displays the test R^2 for each of these models.

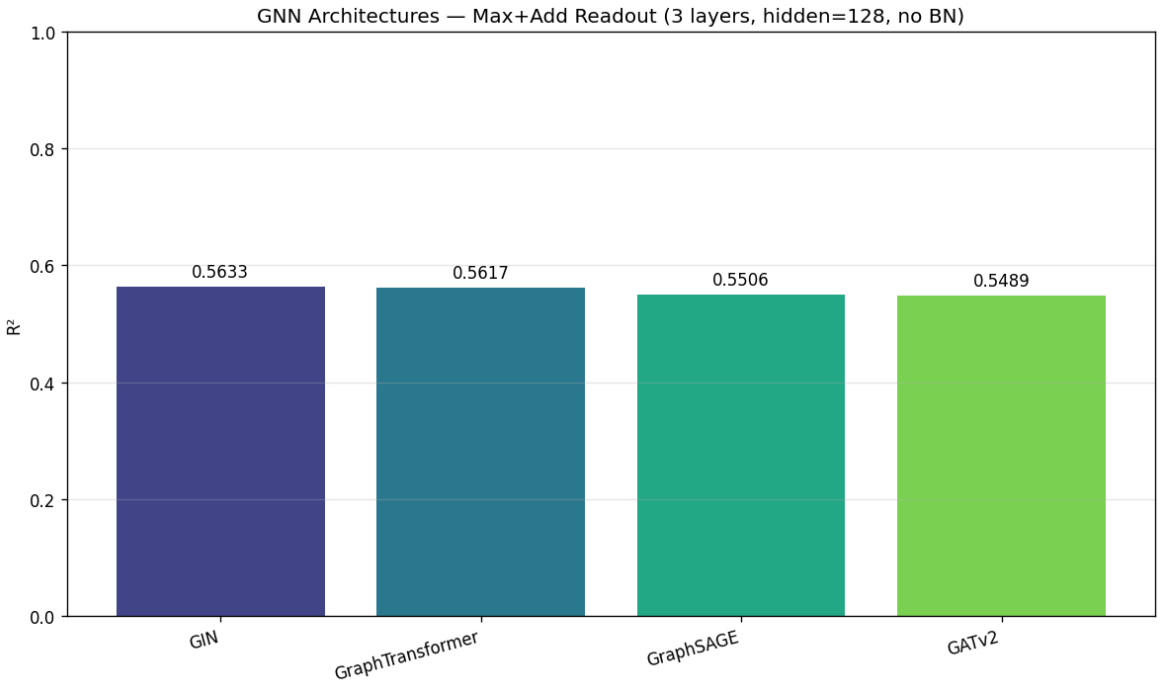


Figure 28: Test R^2 for GNN architectures with Max + Add readout.

Figure 28 reveals that GIN (R^2 = 0.5633) and GraphTransformer (R^2 = 0.5617) outperformed both the GraphSAGE and GATv2 models. The use of attention mechanisms in both the GraphTransformer and the GraphSAGE models seemed to be better at capturing the heterogeneous environments of the molecules within the dataset. The small difference between each of the four models indicates that the choice of GNN architecture may have just as much of an impact on the resulting R^2 as the choice of GNN message passing function.

Table 6: Architecture comparison results.

Architecture	R^2	Rank
GIN	**0.5633**	1st
GraphTransformer	**0.5617**	2nd
GraphSAGE	0.5506	3rd
GATv2	0.5489	4th

Based on these results, **GIN and GraphTransformer** were selected as the two candidates to carry forward into the depth ablation study described in the next section.

4 Stage 3

4.1 Depth Results

Following the protocol described in Section 4.3.1, both GIN and GraphTransformer were evaluated at depths of 4, 5, and 6 layers. The results of these evaluations are presented in Figure 29 and Table 7.

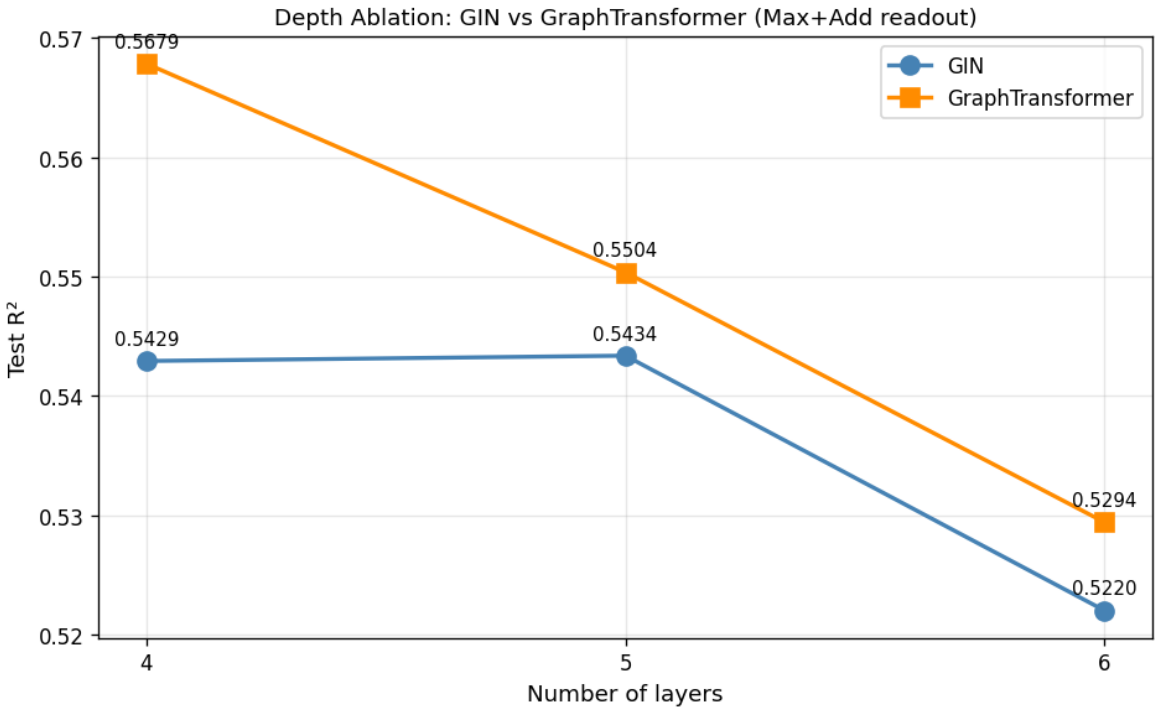


Figure 29: Depth ablation study comparing GIN and GraphTransformer across 4, 5, and 6 message-passing layers using Max + Add readout (no BN).

The results show that GraphTransformer with 4 layers has the highest R^2 of 0.5679, then GraphTransformer with 5 layers has 0.5504, and then GIN with 5 layers achieved 0.5434, but GraphTransformer is better. This indicates a difference from Stage 2 when the GIN model beat GraphTransformer. This result suggests that GraphTransformer is more sensitive to depth, as its performance drops more with each additional layer.

Table 7: Depth ablation results sorted by test R^2.

Model	**Layers**	R^2
GraphTransformer_L4	4	**0.5679**
GraphTransformer_L5	5	0.5504
GIN_ L5	5	0.5434

GIN_ L4	4	0.5429
GraphTransformer_L6	6	0.5294
GIN_L6	6	0.5220

Based on these findings, **Graph Transformer** with 4 layers was chosen as the main GNN architecture to employ in the study of batch normalization placement (Section 4.3.2) before both architectures were considered for inclusion in the final GNN models (Section 4.3.3).

4.2 Batch Normalization Placement Results

Following the depth selection, a batch normalization (BN) placement study was conducted on Graph Transformer-4L as described in Section 4.3.2. Eleven configurations were evaluated, varying where BN layers are inserted within the forward pass: after individual convolutional layers (conv1 through conv5), after all five convolutional layers simultaneously (convs_all), after FC layers only (FC1_only, FC2_only, FC_both), after every layer (BN_all), and a no-BN baseline. All other hyperparameters remained identical across configurations.

Table 8: Batch normalization placement results for GT-4L, sorted by test R^2.

Configuration	BN Location	R^2	RMSE
GT4L_BN_all	**Conv 1–4 + FC1 + FC2**	**0.5906**	**0.4381**
GT4L_BN_FC_both	FC1 + FC2 only	0.5816	0.4429
GT4L_BN_convs_all	Conv 1-4	0.5787	0.4444
GT4L _BN_conv4	Conv 4 only	0.5781	0.4447
GT4L _BN_FC1_ only	FC1 only	0.5769	0.4454
GT4L _BN_conv3	Conv 3 only	0.5767	0.4455
GT4L _BN_FC2_ only	FC1 only	0.5727	0.4476
GT4L _BN_conv1	Conv 1 only	0.5685	0.4489
GT4L _BN_conv2	Conv 2 only	0.5663	0.4509

As shown in Table 8, GT4L_BN_all achieved the highest R^2 of 0.5906 and the lowest RMSE of 0.4381. The R^2 value of the model without Batch Normalization was 0.5679, indicating that BN adds benefit to the model regardless of where it is applied to the model. The model with BN only in the two FC layers achieved the second highest R^2 of 0.5816. BN applied to all layers was chosen as the standard normalization strategy to be utilized in all experiments.

4.3 Final GNN Configuration Results

Following the BN placement study (Section 4.3.2), which identified BN_all as the optimal configuration, GraphTransformer-4L was trained under the final fully regularized settings described in Section 4.3.3: hidden dimension 256, 300 epochs.

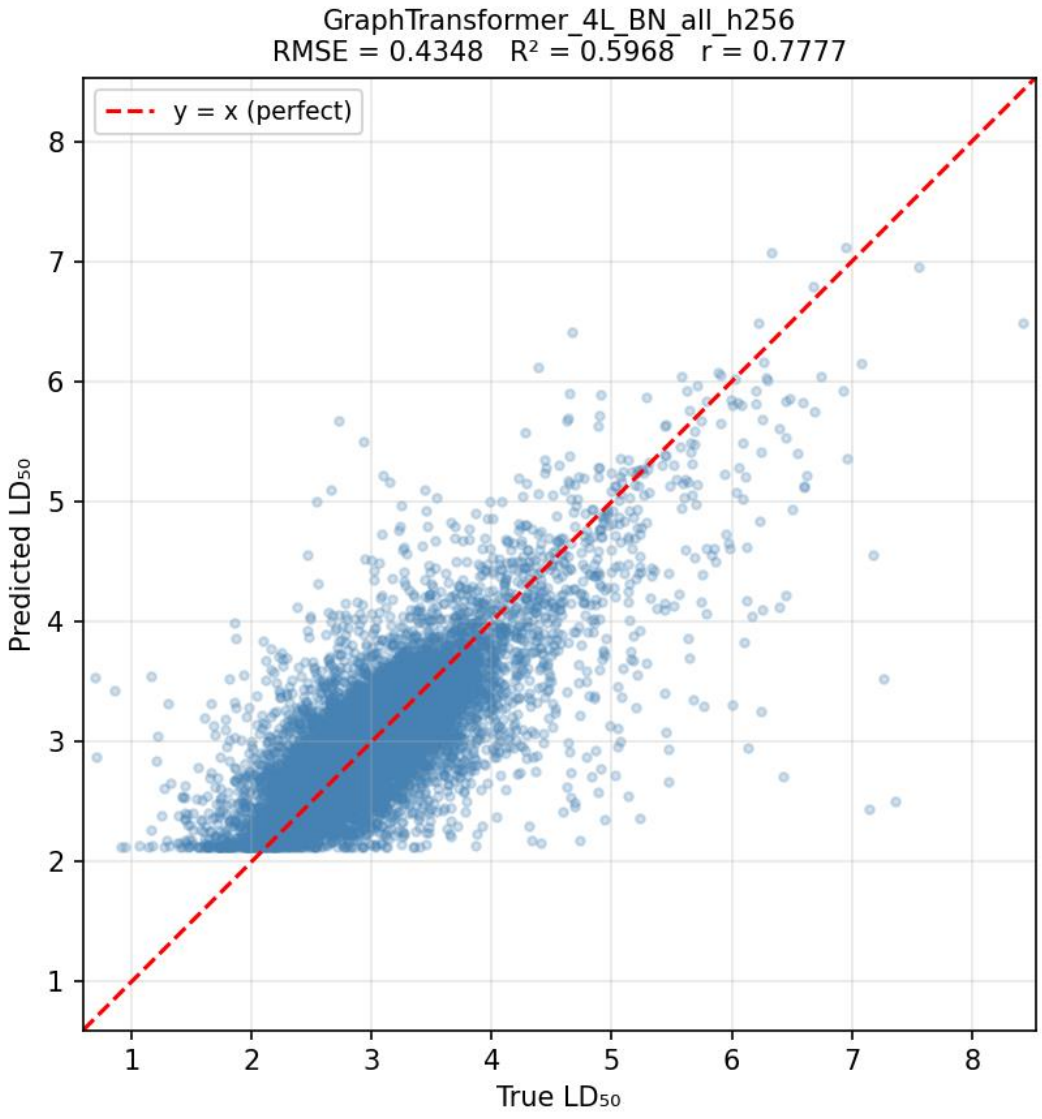


Figure 30: Predicted vs. True LD_{50} scatter plot for GraphTransformer.

Figure 30 and illustrate the improvement of model through the scaling of the hidden dimension and training duration. For the scaling of these two parameters, GraphTransformer achieved an R^2 of 0.5968, an RMSE of 0.4348, and an r value of 0.7777. The scatter plot for model indicate a good correlation between the predicted and observed LD_{50} values, as evidenced by the distribution of the data points along the identity diagonal line.

5 Stage 4 Fingerprint-Based 2D CNN Models

5.1 Individual Fingerprint Results

As described in Section 4.4.2, nine molecular fingerprint types were each evaluated using a 2D CNN model trained under identical conditions. The results of each of these models are provided in Table 9.

Table 9: Performance of nine fingerprint types on the LD_{50} test set using the 2D CNN. Sorted by R^2 descending.

Fingerprint	R^2	RMSE	Pearson r
Avalon	0.5527	0.4579	0.7520
RDK5	0.5295	0.4696	0.7377
ECFC4	0.5291	0.4698	0.7312
ECFP4	0.5225	0.4731	0.7277
RDK6	0.5210	0.4739	0.7294
RDK7	0.5028	0.4828	0.7185
MACCS	0.5016	0.4834	0.7191
FCFC4	0.5009	0.4837	0.7152
FCFP4	0.4952	0.4865	0.7159

Avalon achieved the best individual performance with $R^2 = 0.5527$ and RMSE = 0.4579, followed by RDK5 ($R^2 = 0.5295$) and ECFC4 ($R^2 = 0.5291$). FCFP4 and FCFC4 recorded the lowest R^2 values, while MACCS keys achieved $R^2 = 0.5016$.

5.2 Fingerprint Fusion Results

Following the individual fingerprint evaluation described in Section 4.4.3, the top-3 fingerprints (Avalon, RDK5, and ECFC4) were fused into a 3-channel 2D CNN model as described in Section 4.4.3. Results are shown in Table 10.

Table 10: Comparison of individual fingerprint models and the concatenated 3-channel CNN fusion on the LD_{50} test set.

Model	R^2	RMSE	Pearson r
Avalon (individual)	0.5527	0.4579	0.7520
RDK5 (individual)	0.5295	0.4696	0.7377
ECFC4 (individual)	0.5291	0.4698	0.7312
3-channel CNN (fusion)	0.5624	0.4529	0.7561

The 3-channel fusion model achieved $R^2 = 0.5624$ outperforming all three individual models. This indicates that fusion is better.

6 Stage 5 Physicochemical Descriptor MLP

As described in Section 4.5.1, a plain MLP was trained on the 208 RDKit physicochemical descriptors only, with no graph or fingerprint representation of the molecules. The results are presented in Figures 31 and Table 11.

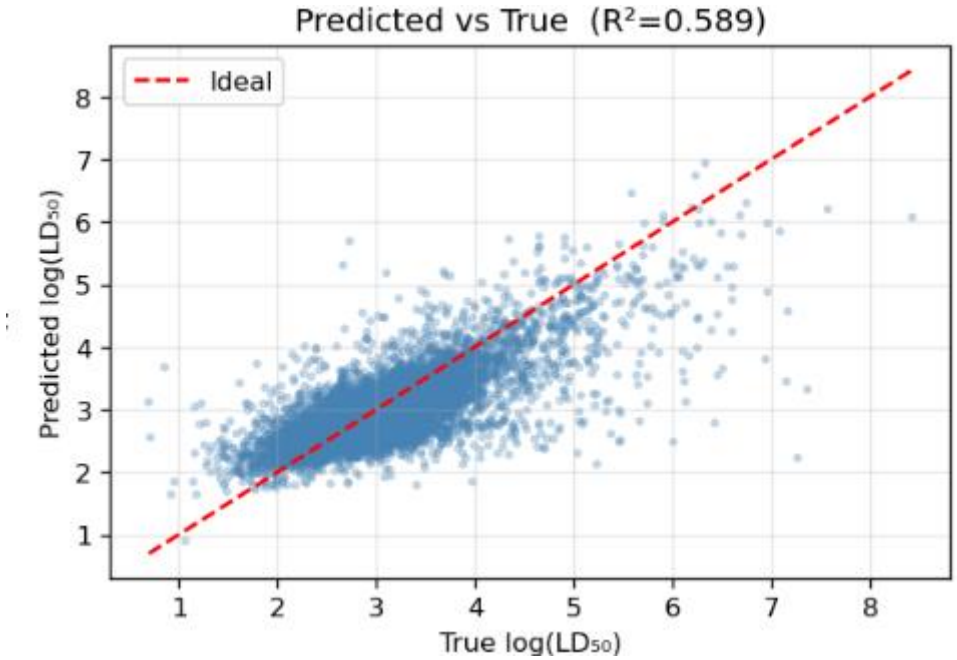


Figure 31: Predicted vs. true log LD_{50} scatter.

Table 11: Final test-set performance of the Descriptor MLP.

Metric	Value
R^2	0.5890
RMSE	0.4389
Pearson r	0.7734

The descriptor MLP achieved an R^2 of 0.5890 and RMSE of 0.4389, outperforming all individual fingerprint models (best: Avalon, R^2 = 0.5527) and the 3-channel fusion model (R^2 = 0.5624).

7 Stage 6 DeepToxiGraph Hybrid Model Results

The DeepToxiGraph model is a hybrid model that takes the three different molecular representations described in Chapter 3. These three representations are the GNN graph embedding (GraphTransformer), the 4096-bit fingerprint representation (Avalon + RDKit + ECFP4+ Atom-Pair), and 208 RDKit descriptors. Each of these representations is passed through separate branches to create three 256-bit embeddings, which are then concatenated and passed through the shared fully connected layer that was described for each of the three branches.

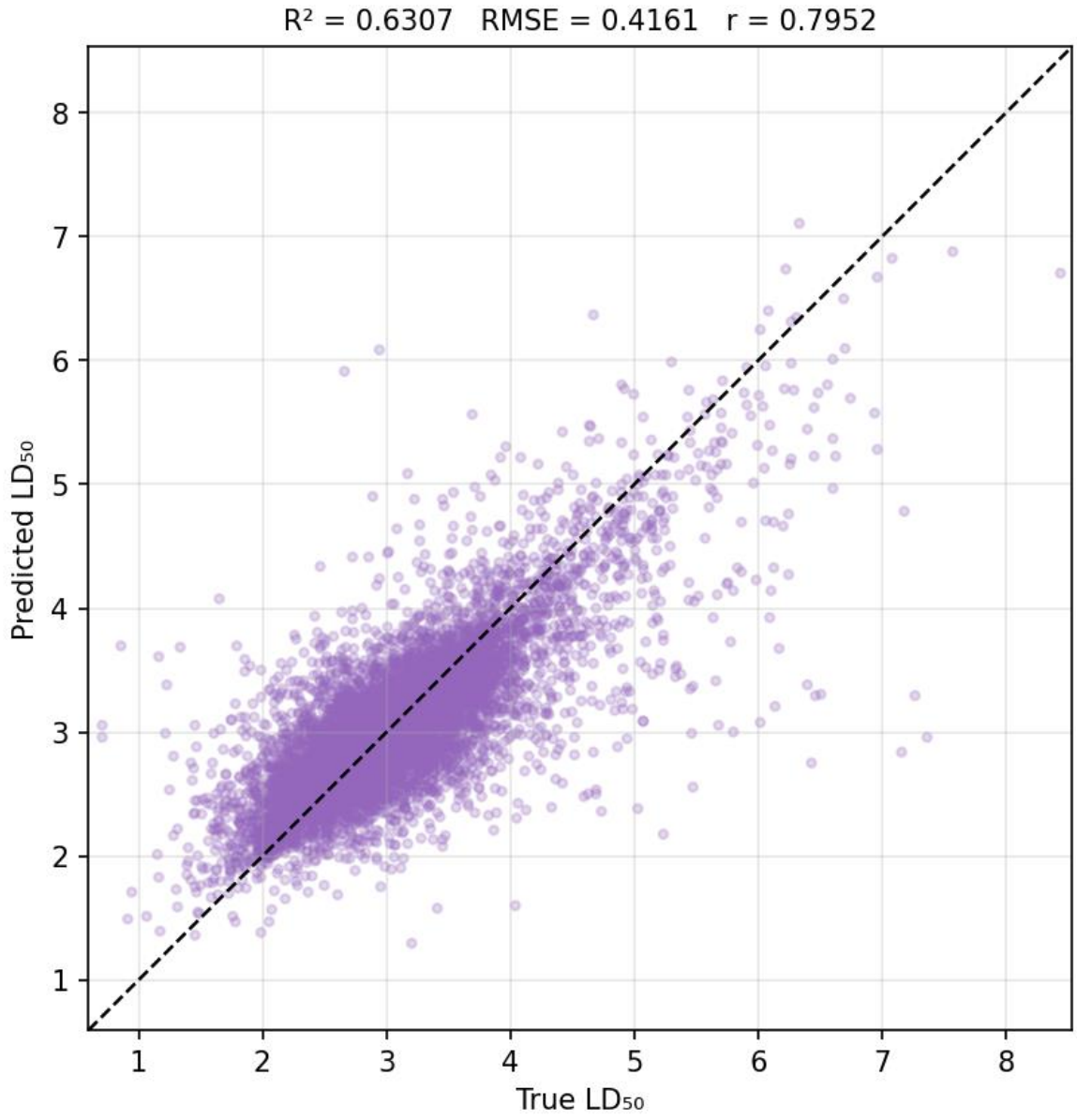


Figure 32: Scatter plot for Hybrid_GT4L (R^2 = 0.6307, RMSE = 0.4161, r = 0.7952).

The results of the hybrid model were better than any of the individual GNN models. Table 12 reports that **Hybrid_GT** model achieved an **R^2 of 0.6307, an RMSE of 0.4161**, and an r of 0.7952. This result for the hybrid model is better than that of the individual GNN models. The scatter plot in Figure 32 indicates that the predictions of the hybrid model are more closely clustered around the identity line and have fewer extreme outliers.

Table 12: Final performance comparison standalone GNN models vs Hybrid.

Model	GNN Branch	R^2	RMSE	r (Pearson)
Hybrid GT	GraphTransformer (4L, h=256)	0.6307	0.4161	0.7952
GraphTransformer standalone	GraphTransformer (4L, h=256)	0.5968	0.4348	0.7777

Comparing the hybrid model to the standalone GraphTransformer, the results indicate that the Hybrid GT model outperforms the standalone model in all three metrics, achieving an R^2 of 0.6307 compared to 0.5968, an RMSE of 0.4161 compared to 0.4348, and a Pearson r of 0.7952 compared to 0.7777. This confirms that the fusion of the three branches adds predictive value beyond what the GNN branch can achieve alone. The relatively modest improvement suggests that the limitation of the model is not the Graph Neural Network itself, but the fusion of the three branches. Creating a smarter fusion mechanism would be the most impactful direction for future development of this work.

8 Comparison with Related Work

To place the performance of DeepToxiGraph in context, Table 13 compares it with previously published models evaluated on the same acute-toxicity LD_{50} task, and where possible on the same TOXRIC mouse intraperitoneal LD_{50} endpoint used in this work. The most directly comparable study is that of Saravanan et al. [35], who applied a Graph Convolutional Network regression model to the same TOXRIC mouse intraperitoneal endpoint and reported a Pearson correlation of 0.76. ToxiGraphNet [49] is a more recent graph neural network trained on a curated LD_{50} set, reporting an R^2 of 0.5959, and the TOXRIC database paper [11] provides Random Forest baselines on this endpoint. Because dataset size, splitting, and pre-processing differ slightly between studies, these numbers are indicative rather than a perfectly controlled benchmark; the comparison is therefore interpreted qualitatively.

Table 13: Comparison of DeepToxiGraph with related LD_{50} prediction models.

Study / Model	Method	Dataset / endpoint	Reported result
Saravanan et al. [35]	Graph Convolutional Network	TOXRIC mouse i.p. LD_{50}	r = 0.76
ToxiGraphNet [49]	GNN (NNConv) regression	Curated LD_{50} set	R^2 = 0.5959
This work (DeepToxiGraph)	**Hybrid GNN + FP + descriptor**	**TOXRIC mouse i.p. LD_{50}**	**R^2 = 0.6307, r = 0.7952**

On the same TOXRIC mouse intraperitoneal LD_{50} endpoint, the DeepToxiGraph hybrid model (R^2 = 0.6307, r = 0.7952) achieves a higher Pearson correlation than the Graph Convolutional Network of Saravanan et al. [35] (r = 0.76), and a higher R^2 than the ToxiGraphNet model [49] (R^2 = 0.5959). Two factors plausibly explain this improvement. First, DeepToxiGraph uses a Graph Transformer encoder rather than a plain graph convolution, which captures longer-range relationships between atoms. Second, and more importantly, the hybrid model fuses three complementary molecular representations — graph embeddings, fingerprint-derived features, and physicochemical descriptors — whereas the compared models rely on a single graph-based view. These results indicate that, on this dataset, combining multiple molecular representations is an effective way to improve LD_{50} regression performance. It should be noted that the studies do not use an identical train/test split, so the comparison establishes that DeepToxiGraph is competitive with and generally outperforms comparable published models rather than constituting a strictly controlled benchmark.

Rank	Molecule ID	True Value	Predicted	|Difference|
1	4042	2,951572	2,951565	0,000006
2	9559	3,160563	3,160524	0,000039
3	617	3,306773	3,306895	0,000122
4	4482	2,643884	2,643751	0,000134
5	337	2,689755	2,689606	0,000149
6	3647	2,583379	2,583537	0,000158
7	7239	2,185804	2,185643	0,000161
8	2401	2,640011	2,639793	0,000218
9	3796	3,254769	3,254539	0,000229
10	6077	2,945106	2,945343	0,000237
11	3472	3,158379	3,158660	0,000281
12	2815	3,241095	3,241405	0,000311
13	5807	3,486924	3,487237	0,000313
14	3733	2,616148	2,615829	0,000319
15	7687	3,011274	3,010912	0,000362
16	7226	2,801928	2,801565	0,000363
17	4664	2,695143	2,694750	0,000393
18	1197	3,223384	3,223779	0,000394
19	4781	2,577399	2,576978	0,000421
20	10097	3,053584	3,053160	0,000424
21	6306	2,996426	2,995921	0,000505
22	8720	2,795672	2,796191	0,000519
23	7987	2,461991	2,461422	0,000569
24	7023	2,579794	2,579223	0,000571
25	4768	2,761109	2,761683	0,000574
26	10269	3,566402	3,566985	0,000582
27	10074	2,630666	2,630025	0,000640
28	1384	2,671300	2,670657	0,000644
29	6173	2,396600	2,395943	0,000657

Figure 33: Hybrid Model: Ranked predictions sorted by ascending absolute error, showing the top 29 most accurate molecule predictions.

The Hybrid model does a great job of successfully predicting the values of the best-ranking molecules within the database. Figure 33 displays the difference between the actual and predicted values for each molecule within the database. The molecule with the highest rank had a difference of only 0.000006 between its actual and predicted molecular value. For all of the other molecules with the highest predicted values, the difference between the actual and predicted values was below 0.001.

For some of the worst performing molecules within the database, however, the difference between the actual and predicted values for those molecules reaches as high as 4.0.

The model excellently predicts the values of the best-ranking molecules within the database, but struggles with other types of molecules within the database altogether.

9 Application

A web application was developed that deploys the DeepToxiGraph hybrid model, allowing any researcher to obtain the predicted LD_{50} of a chemical compound from its SMILES string without writing any code. The Hybrid GT model is deployed within this application, with its weights from the graphtransformer_weights.pt file exported to the ONNX format. The application was built using React, Vite, and TypeScript for the frontend, which runs on Node.js. The backend is implemented in Python. The SMILES strings provided by users are displayed as chemical structures within the web application using the @rdkit/rdkit JavaScript library, which utilizes the same chemistry library as DeepToxiGraph during training. The web application also displays the results of the experiments performed during this study to give context to the LD_{50} values that are provided.

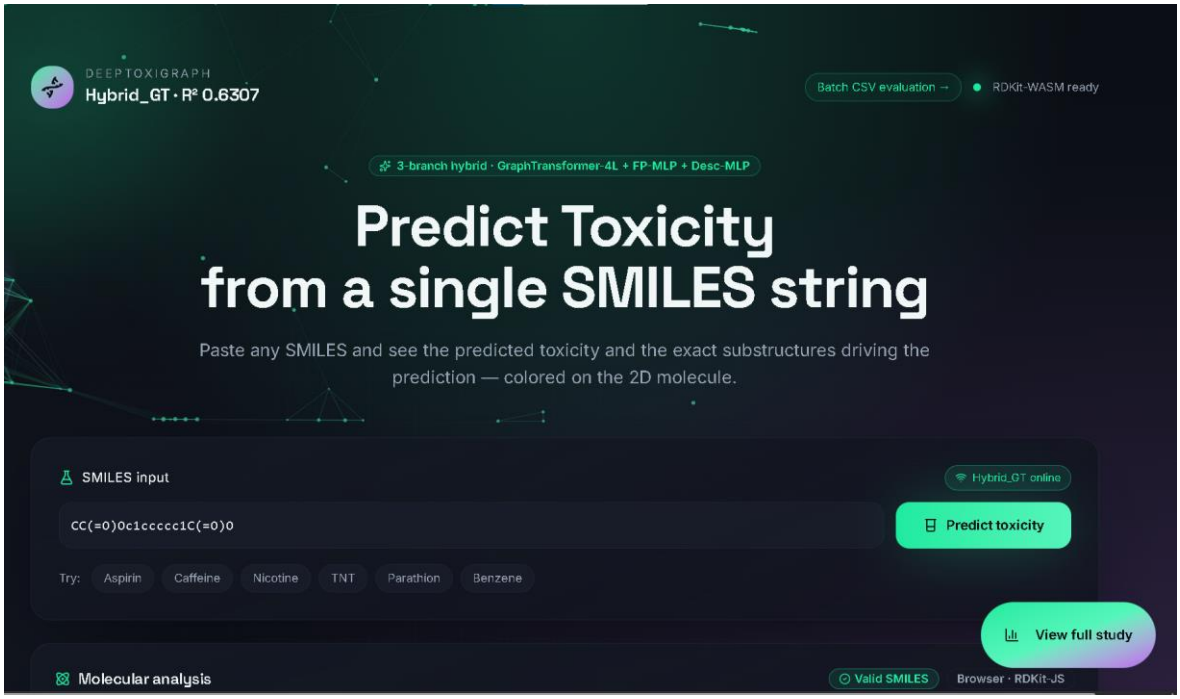


Figure 34: DeepToxiGraph web application interface.

The interface provides a SMILES input field where the user can enter any molecule of their choice or select one of pre-loaded examples, and shows the molecular analysis panel with the canonical SMILES, InChIKey, chemical class, and physicochemical descriptors.

After entering a SMILES string and clicking the "Predict LD_{50}" button, the web site displays the 3D molecular structure, the 2D structure, and the predicted LD_{50} value along with the toxicity classification, as shown in the following figures

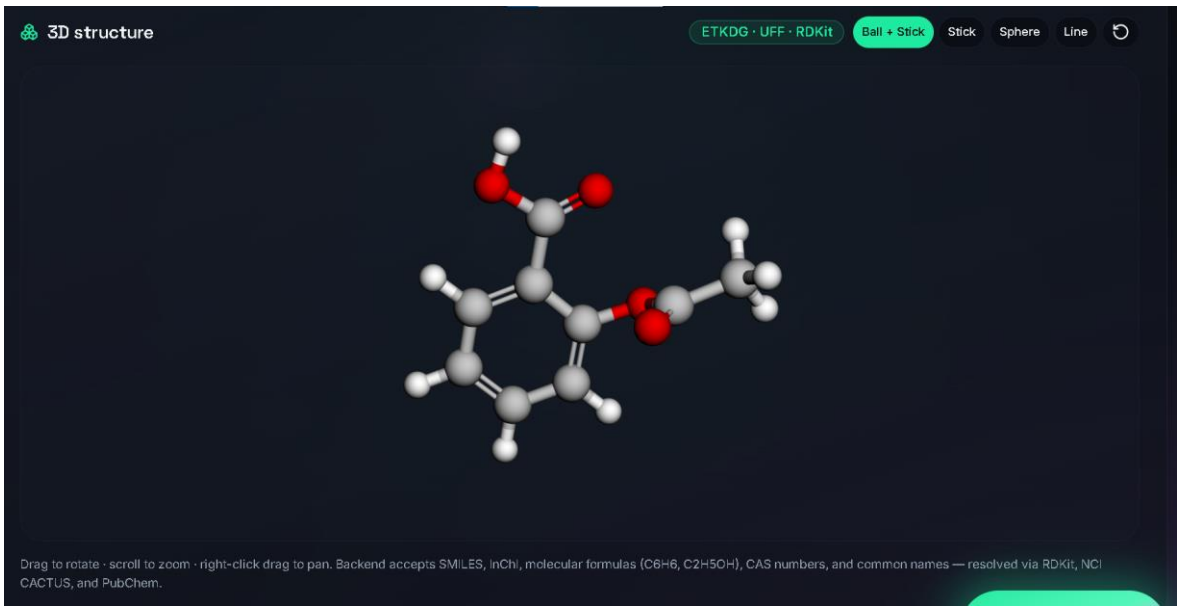


Figure 35: 3D molecular *structure of the input molecule.*

The 3D structure is computed on the Python backend using RDKit ETKDG embedding followed by UFF force-field optimisation. The user can rotate, zoom, and switch between ball-and-stick, stick, sphere, or line rendering modes.

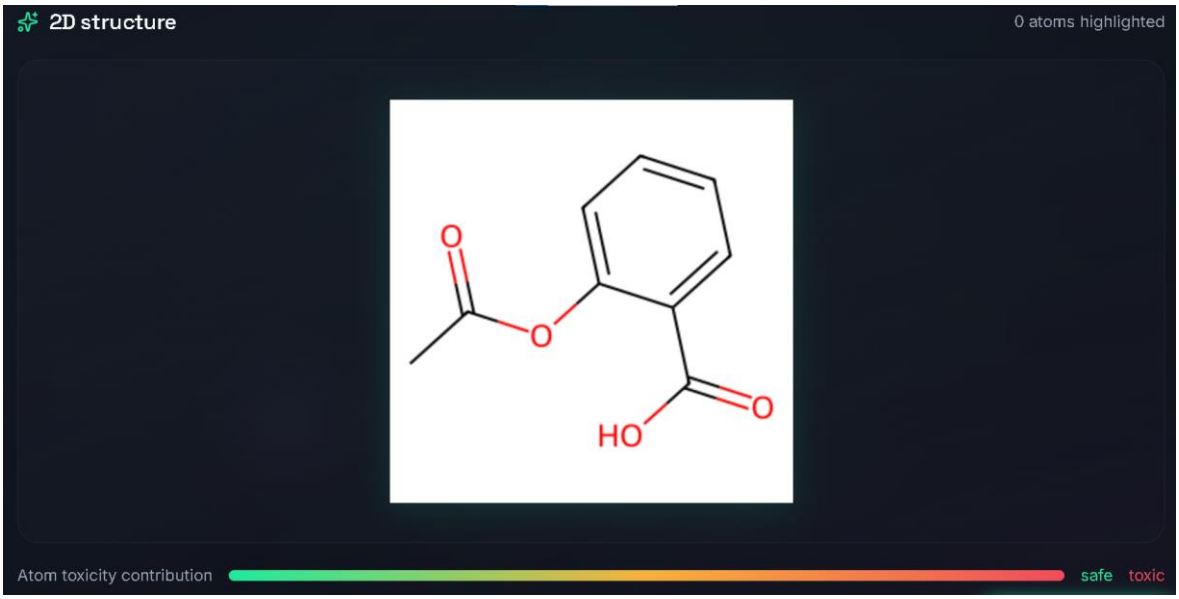


Figure 36: 2D molecular structure of the input molecule.

The 2D structure is rendered directly in the browser from the canonical SMILES using the @rdkit/rdkit JavaScript library, giving a clear schematic of atoms, bonds, and functional groups.

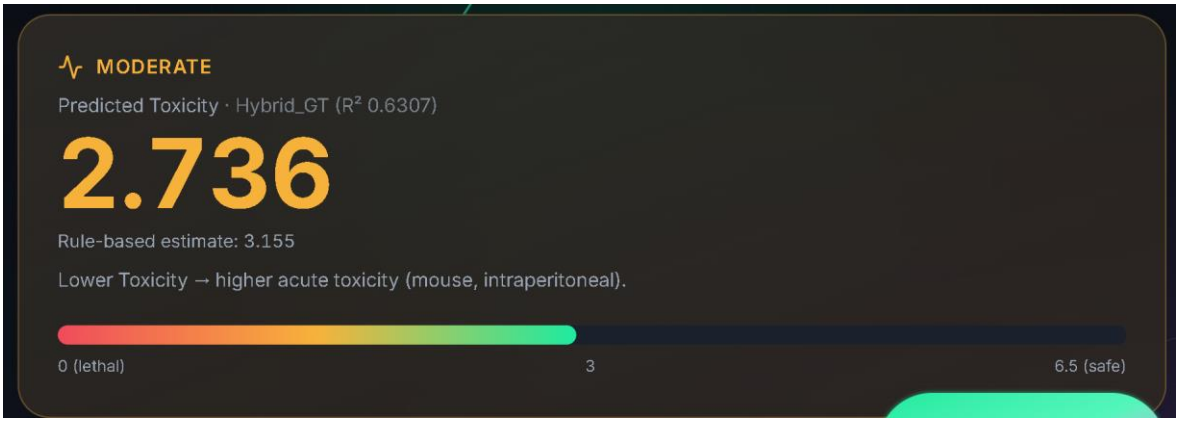


Figure 37: Prediction output of the DeepToxiGraph web site.

The panel reports the predicted -log(mol/kg) LD_{50} value (2.430 for Aspirin), a rule-based estimate, and a toxicity classification (Moderate). A colour-coded scale positions the prediction between low and high toxicity, and the toxicophore panel reports whether any known structural alerts were detected.

The web application also displays the **results of experiments performed during the study**, giving researchers meaningful context for interpreting the predicted LD_{50} values in comparison to known benchmarks.

10 Summary

In this chapter, the results of the experimental stages of the DeepToxiGraph framework are presented. Each of the experiments were performed in stages, with each stage building upon the decisions made confirmed in the previous experiment. Such experiments included testing each of the GNN models, the 2D CNN models that used molecular fingerprints, the MLP that used physicochemical descriptors, and, finally, the test of the hybrid model. The main conclusions that were made from these experiments are as follows:

- The strategies that combined **max and add** pooling outperformed those that used a single pooling method.
- Of the five different GNN that were tested, the best results were seen with both the GraphTransformer and **GIN** models. The **GraphTransformer** model achieved the best performance with **four** layers.
- Increasing the hidden dimension to 256 in the GraphTransformer-4L standalone model, training for 300 epochs with batch normalization and full regularization, increased the R^2 score to 0.5968.
- Within the nine different types of molecular fingerprints that were tested with the 2D CNNs, the Avalon fingerprints were the best at performing individually. When the top three fingerprint types were fused into a 3-channel CNN, the resulting model had an R^2 score of 0.5624, indicating that combining these representations of the molecules led to a better representation of those molecules than using the best fingerprint type ($R^2 = 0.5527$) alone.
- The MLP that used 208 different RDKit descriptors achieved an R^2 score of 0.5890 with the physicochemical descriptors of the molecules, indicating that, overall, these properties were the most important for predicting LD_{50} and outperformed all individual fingerprint models and the 3-channel CNN fusion model.
- The Hybrid model that used the embeddings of the molecules that were represented by the GNNs, as well as the molecular and physicochemical descriptors, achieved the highest value of R^2 and the lowest RMSE of any of the models tested. Specifically, the Hybrid GT model achieved an R^2 score of 0.6307 and an RMSE of 0.4161 on the test set.

- A web application was developed that deploys the DeepToxiGraph hybrid model, allowing researchers to obtain the predicted LD_{50} value of any chemical compound from its SMILES string without writing any code. The application displays the 2D and 3D molecular structures, computes physicochemical descriptors, and outputs the predicted LD_{50} value along with its toxicity level

GENERAL CONCLUSION

General conclusion

The task of predicting the acute toxicity of a molecule based upon its chemical structure is a problem that exists at the intersection of several different fields: chemistry, biology, and machine learning. Deep learning methods have been used to approach this problem over the past decade, moving from the use of handcrafted molecular structure descriptors to learned representations of the molecules. Graph neural networks are perhaps the most natural approach to the problem, as they are essentially models that are applied to the molecule as a graph to determine which aspects of its structure are important for predicting its toxicity.

DeepToxiGraph was built in order to answer two specific questions regarding the field. The first is: Do the different possible architectures for GNNs exhibit different abilities to accurately predict the toxicity of compounds? The second question is, does a model that incorporates both graph representations, fingerprint representations, and descriptor representations lead to better predictions than models that use only one of those representations? Both of these questions were answered through a series of experiments performed on the LD_{50} dataset from the TOXRIC database.

The results of the experiments indicated that the GNN architectures and the placement of batch normalization had an effect on the models' predictive abilities. Graph Transformer models with four message-passing layers and batch normalization after every convolutional and fully-connected layer outperformed the other architectures tested. The Graph Transformer was therefore selected as the GNN branch encoder of the hybrid DeepToxiGraph model. The hybrid model, which simultaneously uses graph, fingerprint, and descriptor representations of each molecule, was found to outperform the best individual models, achieving an R^2 of 0.6307 compared to 0.5968 for the standalone Graph Transformer.

Despite the successful performance of DeepToxiGraph in answering the questions posed, there are also some limitations to the model. For example, DeepToxiGraph was trained only on the mouse intraperitoneal LD_{50} endpoint. Other types of toxicity need to be accounted for, such as hepatotoxicity, mutagenicity, and carcinogenicity. DeepToxiGraph fuses the three representations through a sigmoid-gated concatenation; more sophisticated fusion strategies such as cross-attention may yield further improvements. DeepToxiGraph does not incorporate representations of the three-dimensional structure of the compounds or

their quantum chemical features, these features have been effective within other types of machine learning models for predicting the toxicity of compounds.

Several directions are envisioned for future work. The model could be extended to multi-task learning across the multiple toxicity endpoints available in TOXRIC, allowing a shared molecular representation to transfer information between routes of administration. The GNN branch could be enriched with three-dimensional conformational information, which the current 2D graph representation does not capture. The sigmoid-gate fusion could be replaced by a cross-attention mechanism between the three branches, enabling adaptive weighting per molecule. Explainability methods such as GNNExplainer could be applied to identify the atoms and substructures most responsible for a given LD_{50} prediction. Finally, the methodology could be transferred to other molecular property-prediction tasks such as solubility, lipophilicity, or bioactivity, and integrated into the DeepToxiGraph web site as additional features.

REFERENCES

[1] R. Todeschini and V. Consonni, Molecular Descriptors for Chemoinformatics, vol. 41. Wiley-VCH, Weinheim, Germany, 2009. ISBN: 9783527318520.

[2] D. Nguyen, “Some mathematical perspectives of graph neural networks,” M.Math. thesis, Dept. of Applied Mathematics, University of Waterloo, Waterloo, ON, Canada, 2022. Available: https://uwspace.uwaterloo.ca/items/b7cebd37-c236-4c5a-8089-d64cda9c78e7.

[3] Z. Liu and J. Zhou, Introduction to Graph Neural Networks. Morgan & Claypool Publishers, 2020. DOI: 10.2200/S00980ED1V01Y202001AIM045. Free preprint: https://arxiv.org/abs/2005.01089.

[4] E. Alpaydin, *Introduction to Machine Learning*, 4th ed. Cambridge, MA, USA: MIT Press, 2020.

[5] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” Insights into Imaging, vol. 9, no. 4, pp. 611–629, 2018. DOI: 10.1007/s13244-018-0639-9. PMCID: PMC6108980.

[6] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in Proc. ICML, pp. 1263–1272, 2017. DOI: 10.48550/arXiv.1704.01212.

[7] A. D. White, *Deep Learning for Molecules and Materials*, Living Journal of Computational Molecular Science, vol. 3, no. 1, p. 1499, 2022. DOI: 10.33011/livecoms.3.1.1499. Available at: https://dmol.pub

[8] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," in *AAAI Workshop on Deep Learning on Graphs: Methods and Applications (DLG-AAAI'21)*,2021.DOI:10.48550/arXiv.2012.09699.**Free:** https://arxiv.org/abs/2012.09699

[9] A. Vaswani et al., “Attention is all you need,” in Advances in Neural Information Processing Systems, vol. 30, 2017. DOI: 10.48550/arXiv.1706.03762.

[10] H. Zhu, T. M. Martin, L. Ye, A. Sedykh, D. M. Young, and A. Tropsha, “Quantitative structure–activity relationship modeling of rat acute toxicity by oral exposure,” Chemical Research in Toxicology, vol. 22, no. 12, pp. 1913–1921, 2009. DOI: 10.1021/tx900189p. PMCID: PMC2796713.

[11] L. Wu, B. Yan, J. Han, R. Li, J. Xiao, S. He, and X. Bo, "TOXRIC: a comprehensive database of toxicological data and benchmarks," Nucleic Acids Research, vol. 51, no. D1, pp. D1432–D1445, 2023. DOI: 10.1093/nar/gkac1074.

[12] ECHA, Practical Guide: How to Use and Report QSARs, Version 3.1. European Chemicals Agency, July 2016. Available: https://echa.europa.eu/documents/10162/13655/pg_report_qsars_en.pdf.

[13] L. David, A. Thakkar, R. Mercado, and O. Engkvist, “Molecular representations in AI-driven drug discovery: a review and practical guide,” Journal of Cheminformatics, vol. 12, no. 1, p. 56, 2020. DOI: 10.1186/s13321-020-00460-5. PMCID: PMC7495975.

[14] D. Rogers and M. Hahn, “Extended-connectivity fingerprints,” Journal of Chemical Information and Modeling, vol. 50, no. 5, pp. 742–754, 2010. DOI: 10.1021/ci100050t. PMID: 20426451.

[15] P. Willett, J. M. Barnard, and G. M. Downs, "Chemical similarity searching," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 6, pp. 983–996, 1998. DOI: 10.1021/ci9800211.

[16] C. Hansch and T. Fujita, “ρ-σ-π Analysis. A method for the correlation of biological activity and chemical structure,” Journal of the American Chemical Society, vol. 86, no. 8, pp. 1616–1626, 1964. DOI: 10.1021/ja01062a035.

[17] F. Sahigara, K. Mansouri, D. Ballabio, A. Mauri, V. Consonni, and R. Todeschini, “Comparison of different approaches to define the applicability domain of QSAR models,” Molecules, vol. 17, pp. 4791–4810, 2012. DOI: 10.3390/molecules17054791. PMCID: PMC6268380.

[18] A. Mayr et al., “DeepTox: Toxicity prediction using deep learning,” Frontiers in Environmental Science, vol. 3, p. 80, 2016. DOI: 10.3389/fenvs.2015.00080. PMCID: PMC4737446.

[19] J. M. Stokes et al., “A deep learning approach to antibiotic discovery,” Cell, vol. 180, no. 4, pp. 688–702, 2020. DOI: 10.1016/j.cell.2020.01.021. PMCID: PMC8349178.

[20] D. K. Duvenaud et al., “Convolutional networks on graphs for learning molecular fingerprints,” in Advances in Neural Information Processing Systems, vol. 28, pp. 2224–2232, 2015. DOI: 10.48550/arXiv.1509.09292.

[21] K. Yang et al., “Analyzing learned molecular representations for property prediction,” Journal of Chemical Information and Modeling, vol. 59, no. 8, pp. 3370–3388, 2019. DOI: 10.1021/acs.jcim.9b00237. PMID: 31361481. Free preprint: DOI: 10.26434/chemrxiv.8058240.

[22] Z. Xiong et al., “Pushing the boundaries of molecular representation for drug discovery with graph attention mechanisms,” Journal of Medicinal Chemistry, vol. 63, no. 16, pp. 8749–8760, 2020. DOI: 10.1021/acs.jmedchem.9b00959. PMID: 31408336. Free preprint: DOI: 10.26434/chemrxiv.9743967.

[23] H. Ma, Y. Bian, Y. Rong, W. Huang, T. Xu, W. Xie, G. Ye, and J. Huang, “Multi-view graph neural networks for molecular property prediction,” arXiv:2005.13607, 2020. Available: https://arxiv.org/abs/2005.13607

[24] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. ICLR, 2017. DOI: 10.48550/arXiv.1609.02907.

[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. ICLR, 2018. DOI: 10.48550/arXiv.1710.10903.

[26] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in Proc. ICLR, 2019. DOI: 10.48550/arXiv.1810.00826.

[27] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Advances in Neural Information Processing Systems, vol. 30, pp. 1025–1035, 2017. DOI: 10.48550/arXiv.1706.02216.

[28] G. Landrum, RDKit: Open-Source Cheminformatics Software, 2025. Available: https://www.rdkit.org.

[29] A. Abdo, B. Chen, C. Mueller, N. Salim, and P. Willett, “Ligand-based virtual screening using Bayesian networks,” *Journal of Chemical Information and Modeling*, vol. 50, no. 6, pp. 1012–1020, 2010.

[30] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, “Computational methods in drug discovery,” Pharmacological Reviews, vol. 66, no. 1, pp. 334–395, 2014. DOI: 10.1124/pr.112.007336. PMCID: PMC3880464.

[31] D. Weininger, “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules,” Journal of Chemical Information and Computer Sciences, vol. 28, no. 1, pp. 31–36, 1988. DOI: 10.1021/ci00057a005.

[32] S. Riniker and G. A. Landrum, “Open-source platform to benchmark fingerprints for ligand-based virtual screening,” Journal of Cheminformatics, vol. 5, no. 1, p. 26, 2013. DOI: 10.1186/1758-2946-5-26. PMCID: PMC3686626.

[33] X. Lu, Q. Lu, Z. Li, and X. Yao, “Estimation of acute oral toxicity in rat using local lazy learning,” Journal of Cheminformatics, vol. 6, no. 1, art. 26, 2014. DOI: 10.1186/1758-2946-6-26.

[34] Y. Xu, J. Pei, and L. Lai, “Deep learning based regression and multiclass models for acute oral toxicity prediction with automatic chemical feature extraction,” Journal of Chemical Information and Modeling, vol. 57, no. 11, pp. 2672–2685, 2017. DOI: 10.1021/acs.jcim.7b00244.

[35] K. M. Saravanan, J.-F. Wan, L. Dai, J. Zhang, J. Z. H. Zhang, and H. Zhang, “A deep learning based multi-model approach for predicting drug-like chemical compound’s toxicity,” Methods, vol. 226, pp. 164–175, 2024. DOI: 10.1016/j.ymeth.2024.04.020.

[36] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, 2015. DOI: 10.1016/j.neunet.2014.09.003. PMID: 25462637. Free preprint: arXiv:1404.7828.

[37] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. Free online: https://www.deeplearningbook.org/.

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

[39] Y. Ma and J. Tang, Deep Learning on Graphs. Cambridge University Press, 2021. Available: https://cse.msu.edu/~mayao4/dlg_book/.

[40] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive study on graph neural networks,” IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2021. DOI: 10.1109/TNNLS.2020.2978386. Free preprint: arXiv:1901.00596.

[41] J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI Open, vol. 1, pp. 57–81, 2020. DOI: 10.1016/j.aiopen.2021.01.001. arXiv:1812.08434.

[42] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, “A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future

directions," Journal of Big Data, vol. 11, p. 18, 2024. DOI: 10.1186/s40537-023-00876-4.

[43] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in Proc. ICML, pp. 448–456, 2015. DOI: 10.48550/arXiv.1502.03167.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014. Available: https://jmlr.org/papers/v15/srivastava14a.html.

[45] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. arXiv:1903.02428.

[46] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature," Geoscientific Model Development, vol. 7, no. 3, pp. 1247–1250, 2014. DOI: 10.5194/gmd-7-1247-2014.

[47] N. R. Draper and H. Smith, Applied Regression Analysis, 3rd ed. New York, NY, USA: Wiley-Interscience, 1998.

[48] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in Noise Reduction in Speech Processing, Springer Topics in Signal Processing, vol. 2. Berlin, Germany: Springer, 2009, pp. 1–4. DOI: 10.1007/978-3-642-00296-0_5.

[49] ToxiGraphNet: a graph neural network framework for precise toxicity prediction of drug molecules, Journal of Computer-Aided Molecular Design, vol. 39, art. 683, 2025. DOI: 10.1007/s10822-025-00683-4.