CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Modeling Protein-Ligand Binding Affinity Using Graph Neural Networks: Integrating Molecular Interactions and Physics-Based Properties

Wilson Li and Yuan Wei

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Modeling Protein-Ligand Binding Affinity Using Graph Neural Networks: Integrating Molecular Interactions and Physics-Based Properties

Wilson Li and Yuan Wei

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Wilson Li and Yuan Wei

Academic Supervisor: Alexandre Graell II Amat, Department Electrical Engineering, Chalmers University of Technology
Industrial Supervisor: Lili Cao, Molecular AI, BioPharmaceuticals R&D, AstraZeneca
Examiner: Ola Engkvist, Department Computer Science Engineering, Chalmers University of Technology and Molecular AI, BioPharmaceuticals R&D, AstraZeneca

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2025

Wilson Li and Yuan Wei
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Predicting protein-ligand binding affinity has always been one of the primary challenges in drug discovery. Though many machine learning approaches have been applied and reached substantial progress, despite advancements, the accurate prediction by integrating molecular interactions and physics-based properties of ligands and proteins remains challenging. In this project, we present a graph neural networks (GNNs)-based framework for predicting protein–ligand binding affinity, using publicly accessible CrossDocked2020 dataset. Our project compares three message-passing architectures—Linear, Set Transformer Aggregation (STA), and Graph Attention Network (GAT). Our best model achieves performance of Pearson's $R \approx 0.79$ and Kendall's $\tau \approx 0.58$. We present a practical GNNs-based framework with plausible binding affinity prediction capabilities, designed to effectively differentiate correct poses from incorrect ones.

Keywords: protein–ligand binding affinity, graph neural networks, molecular interaction, machine learning, deep learning, drug discovery

Acknowledgements

We would like to extend our deepest gratitude to AstraZeneca for providing us with this invaluable opportunity to work on an industrially relevant project. This project allowed us to bridge the gap between academic knowledge and real-world industrial challenges, giving us a unique platform to apply theoretical insights in a practical context. We gratefully acknowledge the scientific computing platform for providing the computational resources essential for this research.

Furthermore, we wish to express our sincere thanks to our industrial supervisor, Lili Cao, whose constant guidance and support ensured that our project remained on the right track and reached a successful conclusion. Her expertise in computational chemistry and machine learning was instrumental in overcoming technical challenges and aligning our efforts with real-world industrial expectations.

We are also profoundly grateful to our academic examiner, Ola Engkvist, for his valuable and constructive feedback, which provided critical insights that significantly enhanced the quality of our work. His guidance helped us refine our approach and ensure the project met academic standards.

Finally, thanks to our family members and friends for their encouragement along the way.

Wilson Li, Yuan Wei, Gothenburg, 2025-05-20

Contents

List of Figures

List of Tables

1

Introduction

Proteins carry out a multitude of physiological functions within living organisms, with up to three billion distinct proteins present in human cells [1]. Rather than operating autonomously, proteins require association with other molecules—termed ligands—to fulfill their biological roles [2]. These ligands interact with specialized regions on proteins, often referred to as binding sites, enabling diverse molecular processes [3]. Throughout the binding event, ligands undergo conformational rearrangements to achieve optimal complementarity with the protein binding site. The binding affinity, defined as a quantitative measure of interaction strength, increases in accordance with the stability of the ligand–protein complex [4]. Protein–ligand interactions underpin numerous critical biological phenomena, including antibody–antigen recognition [3], intercellular communication [5] and signal transduction pathways [2]. Aberrant interactions can result in pathological conditions, underscoring the necessity to elucidate the underlying molecular mechanisms in order to facilitate novel therapeutic discovery [3]. Given that conventional drug development is both labor-intensive and time-consuming [6], there exists an urgent demand for efficient and accurate computational approaches to predict protein–ligand affinity, thereby expediting the drug discovery pipeline [7].

The traditional protein–ligand binding affinity prediction methods can be classified into three main categories: physics-based, empirical, knowledge-based approaches. Physics-based methods—such as free energy perturbation (FEP), molecular mechanics Poisson–Boltzmann surface area (MMPBSA) , and molecular mechanics generalized Born surface area (MMGBSA) [8] —employ rigorous thermodynamic calculations to estimate binding free energies with high accuracy but incur substantial computational cost, limiting their suitability for high-throughput screening. Empirical scoring functions decompose interactions into parameterized terms (e.g., van der Waals, electrostatics, solvation) fitted to experimental datasets and are implemented in docking suites like AutoDock, Glide, and GOLD [9]–[12], offering rapid affinity estimates though with limited generalizability and sensitivity to atypical interactions. Knowledge-based potentials derive statistical interaction energies from large structural databases by converting observed interatomic distance frequencies into pseudopotentials (e.g., DrugScore, DFIRE) [13]–[15], resulting in low computational overhead yet suffering from reference-state ambiguities and biases toward frequent interaction motifs. Hence, accurately determining the binding affinities for diverse protein–ligand interactions remains a significant challenge for traditional computational techniques.

Machine learning (ML) has emerged as a pivotal tool which integrates statistical techniques and computational algorithms to build predictive models directly from empirical data. Broadly, ML methodologies are divided into conventional algorithms—such as decision trees, support vector machines (SVMs), and random forests (RFs)—and deep learning (DL) frameworks. Conventional ML models feature relatively simple architectures and low computational overhead, making them well suited for low-dimensional tasks and offering clear interpretability. For example, RF-Score uses counts of protein–ligand atom-type pairs within the binding-site neighborhood as input to a random forest regressor to predict binding affinity [16], whereas ID-Score employs an SVM trained on fifty distinct protein–ligand descriptors [17]. However, these approaches require extensive manual data preprocessing and feature extraction, which can introduce bias and limit their applicability; notably, the oversimplified representation used by RF-Score has been shown to hinder its performance in virtual screening applications [18]. Furthermore, the inherent complexity of protein–ligand interactions—encompassing geometric complementarity, physicochemical properties of residues, and the presence of additional ligands—renders accurate affinity prediction a significant challenge for traditional ML methods reliant on handcrafted features.

Deep learning (DL), as an advanced subset of machine learning (ML), leverages architectures such as deep fully connected networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) to model complex patterns in data. Unlike conventional ML approaches that often require handcrafted features, DL models inherently learn hierarchical feature representations directly from raw input, enabling simultaneous feature extraction and prediction. This capability becomes particularly advantageous when processing large-scale datasets, where DL models consistently outperform traditional ML due to their scalability and expressiveness. Owing to their robustness in identifying abstract features and recognizing intricate patterns, DL models have achieved transformative impacts across a variety of domains [19], [20]. In the field of bioinformatics, one of the most notable breakthroughs came from DeepMind's AlphaFold2, which utilized DL-based techniques to significantly advance the prediction accuracy of protein tertiary structures, reshaping the landscape of structural biology [21], [22]. Given these promising developments, DL models are increasingly being adopted for protein–ligand binding affinity prediction. Nevertheless, current DL-based affinity predictors still face several challenges, including suboptimal feature representations, and poorly designed model architectures. Addressing these limitations is essential for further enhancing the accuracy and reliability of affinity prediction models.

In recent years, deep learning has emerged as a promising approach for predicting protein–ligand binding affinity, leveraging different data representations such as amino acid or SMILES sequences [23], 3D voxelized grids [24], and molecular graphs [25]. Among these, graph-based representations combined with graph neural networks (GNNs) have demonstrated strong performance across various tasks in computational biology and chemistry [26], including early explorations in affinity prediction scenarios. One such early application is the ACNN model—a convolutional neural network architecture using graph convolution operations as a comparative baseline [27].

Their approach builds upon the foundational graph convolution technique, which transforms molecular graphs into learned feature spaces through message passing between atoms [28]. Another notable model, PotentialNet, employs a graph neural network framework specifically designed to predict molecular properties, including protein–ligand binding affinities [29]. In contrast to traditional molecular graphs that consider only covalent bonds, PotentialNet also integrates spatial proximity information by incorporating Euclidean distances between atoms when constructing graph edges. This hybrid edge definition allows the model to better capture both chemical connectivity and 3D structural relationships essential for accurate affinity prediction.

Although GNNs have demonstrated remarkable prowess in capturing complex graph-structured representation of molecular systems, only a few studies have incorporated explicit physical interaction terms and physics-based molecular properties into binding-affinity predictors. Herein, we propose a unified framework, *Modelling Ligand Binding Affinities Using GNNs: Integrating Molecular Interactions and Physics-Based Properties*, which (i) adopts a hybrid molecular graph representation encoding both connectivity and three-dimensional geometry, (ii) augments node and edge features with quantitative descriptors of bond strengths, electrostatic potentials, and solvation energies, and (iii) employs a multi-scale message-passing architecture to capture local and long-range intermolecular interactions. We validate our approach on established binding-affinity benchmarks, demonstrate significant gains in predictive accuracy over baseline GNN models, and analyze the impact of each physics-inspired feature on model performance, thereby illustrating the value of integrating chemical physics into deep graph learning for drug discovery.

2

Theory

2.1 Graph Neural Networks

2.1.1 Background

Graph Neural Networks (GNNs) constitute a versatile class of neural models designed to operate on graph-structured data. In contrast to traditional neural networks that assume independent and identically distributed inputs, GNNs are capable of capturing complex interdependencies among entities by treating data as nodes connected by edges[30].

At the core of many modern GNN architectures is the message passing scheme. In its most basic form, a GNN layer updates each node's hidden representation by aggregating information from its neighbors. Formally, consider a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes, and $\mathcal{E}$ the set of edges. Let the feature vector of node i at layer l be denoted by $h_i^{(l)}$. Then, the update rule for node i at the subsequent layer $l+1$ can be expressed as:

$$h_i^{(l+1)} = \texttt{UPDATE}^{(l)}\left(h_i^{(l)}, \texttt{AGGREGATE}^{(l)}\left(\{\, m_{ij}^{(l)} : j \in \mathcal{N}(i)\,\}\right)\right), \tag{2.1}$$

where:

- $h_i^{(l)}$ is the current feature vector of the node i,
- $\mathcal{N}(i)$ denotes the set of neighboring nodes of i,
- $m_{ij}^{(l)}$ is the message transmitted from node j to node i at layer l,
- $\texttt{AGGREGATE}^{(l)}$ is a permutation invariant function (e.g., sum, mean, or max) that consolidates the incoming messages,
- $\texttt{UPDATE}^{(l)}$ is typically a differentiable function, often parameterized by neural networks, which integrates the node's current state with the aggregated message.

A common instantiation of the message function is given by:

$$m_{ij}^{(l)} = \phi\left(h_i^{(l)}, h_j^{(l)}, e_{ij}\right), \tag{2.2}$$

where:

- e_{ij} comprises the features associated with the edge between nodes i and j,
- ϕ is a learnable function, often implemented as a multilayer perceptron (MLP), which models the interaction between the node features and the edge attributes.

In many cases, the entire update is performed through a concatenation of the node's current state and the aggregated message, followed by a linear transformation and a non-linear activation. This can be formalized as:

$$h_i^{(l+1)} = \sigma\left(W^{(l)} \cdot \left[h_i^{(l)} \,\|\, \texttt{AGGREGATE}^{(l)}\left(\{\, m_{ij}^{(l)} : j \in \mathcal{N}(i)\,\}\right)\right]\right), \tag{2.3}$$

where:

- $W^{(l)}$ is a learnable weight matrix for layer l,
- σ denotes an element-wise non-linear activation function (e.g., ReLU),
- $\|$ represents the concatenation operator.

This message passing framework endows GNNs with the ability to capture local and global dependencies by stacking multiple layers. With each successive layer, nodes incorporate information from progressively distant nodes in the graph, thereby enriching their representations. This multihop information aggregation is one of the principal reasons behind the expressive power of GNNs.

Furthermore, the choice of the aggregation function is critical. For example, simple aggregation methods like summation naturally capture the cumulative influence of neighboring nodes, whereas functions like averaging or applying a nonlinear projection may confer additional robustness by normalizing the contributions of nodes with different degrees.

In summary, the underlying mechanism of message passing and aggregation in GNNs provides a fundamentally robust framework for learning from graph data. By directly modeling the relational structure, GNNs efficiently combine local feature interactions with global context, thereby extending the representational capacity of classical neural network architectures to complex and discrete data structures.

2.1.2 Applications in Biochemical Systems

The message passing mechanism plays a crucial role in modeling biochemical systems using Graph Neural Networks (GNNs). For example, in the study of protein-ligand interactions, nodes can represent atoms within a ligand or amino acid residues within a protein. Through message passing, each node iteratively exchanges information with its neighboring nodes, progressively refining its own representation. This process allows nodes to incorporate not only their intrinsic features but also contextual information from their surrounding environment.

This iterative exchange of information enables GNNs to capture the rich and subtle dependencies inherent in biochemical systems. It empowers the model to recognize structural motifs, interaction patterns, and long-range effects that are essential for accurate predictions of binding activities or the identification of key interaction sites.

By stacking multiple layers of message passing, the model accumulates multi-scale information, building detailed and context-aware embeddings for each node. These refined representations form a solid foundation for downstream prediction tasks, such as estimating binding affinity or identifying crucial interaction hotspots.

2.2 Set Transformer Aggregation

Instead of relying on fixed aggregation methods such as summation or averaging, adaptive and differentiable readout mechanisms have been introduced to address the limitations of traditional graph-level pooling strategies and to yield richer, more expressive graph representations [31].

Given a node embedding matrix $H \in \mathbb{R}^{M \times D}$ produced by the final GNN layer (where M is the number of nodes and D is the node feature dimension), the STA readout computes a graph-level representation $\mathbf{h}_G$ through a sequence of attention-based operations:

$$\mathbf{h}_G = \frac{1}{K} \sum_{k=1}^{K} \text{DECODER}_k \left(\text{ENCODER}(H) \right) \tag{2.4}$$

The encoder is defined as a stack of multi-head attention blocks:

$$\text{ENCODER}(H) = \text{MAB}_n(H, H) \tag{2.5}$$

The decoder is built using a learned seed vector $s \in \mathbb{R}^{1 \times D}$, which attends over the encoded node set:

$$\text{DECODER}(Z) = \text{FF} \left(\text{MAB}_m \left(\text{PMA}(Z), \text{PMA}(Z) \right) \right) \tag{2.6}$$

where $\text{PMA}(Z) = \text{MAB}(s, \text{FF}(Z))$ denotes pooling by multihead attention, and each MAB is a multi-head attention block followed by layer normalization and residual connections:

$$\text{MAB}(X, Y) = X + \text{MultiHead}(X, Y, Y) \tag{2.7}$$

By learning to weigh and integrate node features dynamically, the STA model captures more nuanced structural information than standard readouts. Its effectiveness has been demonstrated across over 40 benchmark datasets from various domains, consistently outperforming baseline pooling methods—particularly as the number of message-passing steps increases.

2.3 Graph Attention Networks

Graph Attention Networks (GATs) represent a significant advancement in graph neural network techniques by introducing learnable attention mechanisms for weighting the influence of neighboring nodes during feature aggregation. Unlike standard graph convolution operations that typically treat all neighbors equally, GATs leverage adaptive coefficients to focus on the most relevant information in each node's neighborhood.

2.3.1 Theoretical Foundations of GAT

Given a graph $G = (V, E)$ with node features $\mathbf{h}_i \in \mathbb{R}^F$, the input features are first linearly transformed:

$$\mathbf{h}'_i = \mathbf{W}\mathbf{h}_i, \tag{2.8}$$

where $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is a learnable weight matrix.

An attention score e_{ij} is then computed between node i and each of its neighbors $j \in \mathcal{N}_i$ (including itself) via a shared mechanism a:

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]\right), \tag{2.9}$$

where $\|$ denotes concatenation and $\mathbf{a} \in \mathbb{R}^{2F'}$ is a learnable weight vector.

These attention coefficients are normalized using the softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}. \tag{2.10}$$

The output of the layer is a weighted sum of neighbor features followed by a nonlinearity σ:

$$\mathbf{h}_i^{\text{out}} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right). \tag{2.11}$$

To increase model capacity and training stability, GATs often use multi-head attention. In intermediate layers, outputs from K attention heads are concatenated:

$$\mathbf{h}_i^{\text{out}} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j\right). \tag{2.12}$$

In the final prediction layer, it is common to average the outputs from all heads, rather than concatenate them:

$$\mathbf{h}_i^{\text{out}} = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j\right). \tag{2.13}$$

This attention-based aggregation allows the model to learn both the content and structural importance of each node's neighborhood. By assigning different weights to different neighbors, the network becomes more expressive than fixed-weight methods.[32]

2.3.2 Implementation in graph-structured data

In our framework, GATs are employed as the primary intra-graph message-passing modules within a GNNs architecture modeling protein and ligand graphs. By integrating GATs, our model leverages the adaptive weighting of neighborhood information, allowing each node to selectively focus on the most informative neighbors.

The key advantages of using GATs in our setting include:

- **Adaptive Aggregation:** Unlike conventional methods that equally weight all neighboring nodes, GATs compute attention coefficients that enable nodes to assign variable importance to each neighbor. This selective emphasis is particularly valuable in heterogeneous networks where interactions vary in relevance.
- **Enhanced Representational Power:** The attention mechanism inherent in GATs allows the model to capture subtle structural nuances by learning from multiple subspaces through multi-head attention. This leads to more robust and expressive node representations.
- **Scalability and Flexibility:** GATs are naturally suited for graphs with irregular structures and can be applied inductively to previously unseen nodes or graphs. Their design supports efficient parallelization, making them amenable to both large-scale and sparse data scenarios.
- **Improved Stability:** The multi-head attention mechanism, where several independent attention processes are executed in parallel, helps stabilize training. This diversification in representation learning mitigates overfitting and provides resilience against noise inherent in complex biological networks.

By incorporating these dynamic attention mechanisms, our framework is well-equipped to integrate and refine information from both protein and ligand domains. The adaptive, multi-faceted aggregation facilitated by GATs enhances the capture of intricate intra-graph dependencies, which is essential for accurately modeling biological interactions and ultimately improving downstream predictive performance.

The incorporation of GATs into our framework exemplifies an effective solution to the challenges of processing graph-structured data. By enabling localized, context-dependent weighting of neighboring nodes, GATs contribute a critical adaptive capacity, fostering improved representation and better alignment with the complex, heterogeneous nature of biological networks.

2.4 Machine Learning Training Straegy

2.4.1 Description of Round-Robin Training

In many real-world machine learning applications, the full training set may be too large to fit in memory at once or may be naturally partitioned across multiple files or data sources [33], [34]. The *round-robin* training strategy addresses these challenges by cycling through disjoint data partitions (“chunks”) in a fixed order, drawing one

mini-batch at a time from each, and performing parameter updates incrementally. Below we present a detailed account of this procedure, including notation, algorithmic pseudocode, a concrete numerical example, and cost analysis.

2.4.1.1 Algorithm Description

Here is the notation setup used throughout this algorithm.

- K be the number of data chunks (or sources), indexed $k = 1, \ldots, K$.
- D_k denote the k-th chunk, containing $|D_k|$ training examples.
- B be the mini-batch size (assumed uniform across chunks).
- E be the total number of epochs.
- θ_t be the model parameters after t updates.
- η_t be the learning rate at update step t.
- $\ell(\theta; x, y)$ be the loss on a single example (x, y).

Before training begins, we initialize

$$\theta_0 \sim \text{RandomInit} \quad \text{and} \quad t \leftarrow 0.$$

At a high level, each epoch consists of K sequential passes—one over each chunk—where each pass iterates through all mini-batches in that chunk. The fixed order $1, 2, \ldots, K$ constitutes the "round-robin" schedule. The procedure refers to Algorithm 1.

Algorithm 1 Round-Robin Mini-Batch Training

Require: Data chunks $\{D_k\}_{k=1}^{K}$, epochs E, batch size B
Initialize $\theta_0 \leftarrow$ RandomInit
$t \leftarrow 0$
for $e \leftarrow 1$ to E **do** ▷ Epoch loop
  for $k \leftarrow 1$ to K **do** ▷ Round-robin over chunks
    Create iterator $\mathcal{I}_k \leftarrow \text{Batches}(D_k, B)$
    while $\mathcal{I}_k$.hasNext() **do** ▷ Iterate mini-batches
      $(X, Y) \leftarrow \mathcal{I}_k$.next()
      $g_t \leftarrow \frac{1}{B} \sum_{(x,y)\in(X,Y)} \nabla_\theta \ell(\theta_t; x, y)$
      $\theta_{t+1} \leftarrow \theta_t - \eta_t \, g_t$
      $t \leftarrow t + 1$
    end while
  end for
end for
Ensure: Learned parameters θ_t

Here is a concrete example for round-robin training process. Suppose $K = 3$ data chunks with sizes $|D_1| = 1200$, $|D_2| = 800$, $|D_3| = 2000$ and mini-batch size $B = 200$.

Then:

$$\text{Batches in } D_1 = 6, \quad D_2 = 4, \quad D_3 = 10.$$

For $E = 2$ epochs, the total number of gradient updates is

$$E \times \sum_{k=1}^{3} \left\lceil |D_k|/B \right\rceil = 2 \times (6 + 4 + 10) = 40.$$

The update order is:

$$D_1 : (1-6),\ D_2 : (1-4),\ D_3 : (1-10),\ D_1 : (1-6),\ D_2 : (1-4),\ D_3 : (1-10).$$

2.4.1.2 Memory and Computational Cost Analysis

The round-robin training strategy requires at most one data chunk plus a single mini-batch to reside in memory at any time, i.e. a footprint of $\max_k |D_k| + B$ examples, in contrast to the full dataset size $\sum_k |D_k|$ if all data were loaded simultaneously. Data are read sequentially from each chunk—allowing for efficient buffering or prefetching—and thus avoid the overhead of random shuffling across the entire dataset. Finally, each epoch entails exactly

$$\sum_{k=1}^{K} \left\lceil |D_k|/B \right\rceil$$

gradient updates, corresponding to the sum of mini-batches processed in every chunk.

Overall, round-robin training achieves a balance between memory efficiency and fair data exposure, making it particularly suitable for large-scale, distributed, or streaming machine learning applications.

2.4.2 Description of Interleaved training

Interleaved training is a strategy for learning from multiple data sources or tasks by alternating mini-batches drawn from each source in a finely mixed ("interleaved") schedule [35], [36]. Unlike round-robin, which processes one entire chunk before moving on, interleaved training selects a mini-batch from a different source at every update step, ensuring maximal mixing and continuous exposure to all sources throughout each epoch. This approach is particularly effective in multi-task learning, domain adaptation, and streaming data scenarios where rapid adaptation to each source is desired.

2.4.2.1 Algorithm Description

Here is the notation setup used throughout this algorithm.

- K be the number of data sources (or tasks), indexed $k = 1, \ldots, K$.
- D_k denote the k-th source, containing $|D_k|$ examples.
- B be the mini-batch size per source (assumed equal for simplicity).

- E be the total number of epochs.
- θ_t be the model parameters at update step t.
- η_t be the learning rate at step t.
- p_k be the probability of selecting source k for each mini-batch.

Before training, initialize

$$\theta_0 \sim \text{RandomInit}, \quad t \leftarrow 0, \quad \{p_k\}_{k=1}^{K} \text{ such that } \sum_{k=1}^{K} p_k = 1.$$

At each update step within an epoch, the algorithm samples a source according to the distribution $\{p_k\}$, draws one mini-batch from that source, and updates the model. This yields an interleaving at the granularity of single mini-batches. The procedure refers to Algorithm 2.

Algorithm 2 Interleaved Mini-Batch Training

Require: Data sources $\{D_k\}_{k=1}^{K}$, epochs E, batch size B, sampling weights $\{p_k\}$
Initialize $\theta_0 \leftarrow$ RandomInit
$t \leftarrow 0$
Precompute iterators $\mathcal{I}_k \leftarrow \text{Batches}(D_k, B)$ for each k
for $e \leftarrow 1$ to E **do** ▷ Epoch loop
  repeat ▷ Until all sources are exhausted
    Sample $k \sim \text{Categorical}(p_1, \ldots, p_K)$
    if $\mathcal{I}_k$.hasNext() **then**
      $(X, Y) \leftarrow \mathcal{I}_k$.next()
      $g_t \leftarrow \frac{1}{B} \sum_{(x,y)\in(X,Y)} \nabla_\theta \ell(\theta_t; x, y)$
      $\theta_{t+1} \leftarrow \theta_t - \eta_t \, g_t$
      $t \leftarrow t + 1$
    else
      Remove source k from sampling pool; renormalize $\{p_k\}$
    end if
  until all $\mathcal{I}_k$ are exhausted
end for
Ensure: Learned parameters θ_t

Here is a concrete example for interleaved training process. Consider $K = 3$ sources with sizes $|D_1| = 1000$, $|D_2| = 500$, $|D_3| = 1500$, batch size $B = 250$, and uniform sampling $p_k = \frac{1}{3}$. Each source yields:

$$\text{Batches in } D_1 = 4, \; D_2 = 2, \; D_3 = 6.$$

Within one epoch, updates proceed in a random interleaved order such as:

$$D_2(1), \; D_1(1), \; D_3(1), \; D_1(2), \; D_2(2), \; D_3(2), \; D_3(3), \; D_1(3), \ldots$$

until all 12 total mini-batches have been consumed. Over $E = 2$ epochs, 24 gradient updates occur with rich intermixing of sources at every step.

2.4.2.2 Memory and Computational Considerations

In terms of resource utilization, the memory footprint during interleaved training is confined to a single mini-batch in addition to minimal buffering for each source iterator. Depending on the data source, the input/output pattern may involve either random-access or sequential streaming reads; moreover, the interleaving strategy often requires maintaining multiple open file handles or employing asynchronous prefetching to ensure data availability. From a computational perspective, interleaved training performs exactly

$$\sum_{k=1}^{K} \left\lceil |D_k|/B \right\rceil$$

parameter updates per epoch, matching the update count of round-robin training while providing a finer-grained intermixing of data sources.

Interleaved training maximizes simultaneous exposure to all data sources at the finest granularity, promoting rapid adaptation and balanced learning across heterogeneous datasets or tasks.

2.5 Learning Rate Scheduling

The choice of learning rate is pivotal in the training dynamics of deep learning models, significantly influencing both convergence and final performance. Static learning rates often lead to suboptimal outcomes: an excessively high rate may induce divergence or oscillatory behavior around minima, whereas an overly low rate can result in slow convergence or entrapment within poor local minima. To mitigate these issues, dynamic learning rate adjustment strategies—commonly referred to as *learning rate schedulers*—are widely employed.

The fundamental principle underlying learning rate scheduling is to utilize a relatively high learning rate during the initial phase of training. This approach facilitates rapid initial progress, allowing the optimizer to traverse the parameter space and approach regions that potentially host optimal solutions. As training progresses, the learning rate is systematically reduced, thereby permitting finer adjustments to the model parameters. This reduction, often termed *learning rate annealing*, enhances the stability of convergence and improves the eventual performance of the model.

In this work, two distinct learning rate scheduling strategies are implemented and compared: Cosine Annealing with Warm Restarts and the Cyclical Learning Rate (CLR) with the Triangular2 policy. Both strategies are realized within the PyTorch framework.

2.5.1 Cosine Annealing Warm Restarts Learning Rate Scheduling

The `CosineAnnealingWarmRestarts` scheduler decreases the learning rate following a cosine function over each cycle, and subsequently *restarts* the rate to a higher value once the cycle concludes. This mechanism enables the optimizer to escape

shallow local minima, which is particularly advantageous in tasks characterized by non-convex loss landscapes, such as binding affinity prediction using GNNs.

The following pseudocode (Algorithm 3) provides a detailed explanation of the Cosine Annealing Warm Restarts Learning Rate Scheduling:

Algorithm 3 Cosine Annealing with Warm Restarts

Require: Maximum learning rate $\eta_{\max}$, minimum learning rate $\eta_{\min}$, initial cycle length T_0, cycle multiplier T_{mult}, total number of iterations N

$T_{\text{cur}} \leftarrow 0$
$T_i \leftarrow T_0$
for $t \leftarrow 1$ to N **do**
    Compute learning rate:

$$lr \leftarrow \eta_{\min} + \frac{\eta_{\max} - \eta_{\min}}{2}\left(1 + \cos\left(\pi \frac{T_{\text{cur}}}{T_i}\right)\right)$$

    Update optimizer with lr
    if $T_{\text{cur}} == T_i$ **then** ▷ End of current cycle
        $T_{\text{cur}} \leftarrow 0$
        $T_i \leftarrow T_i \cdot T_{\text{mult}}$ ▷ Extend cycle length
    else
        $T_{\text{cur}} \leftarrow T_{\text{cur}} + 1$
    end if
end for

Ensure: Final learning rate schedule applied.

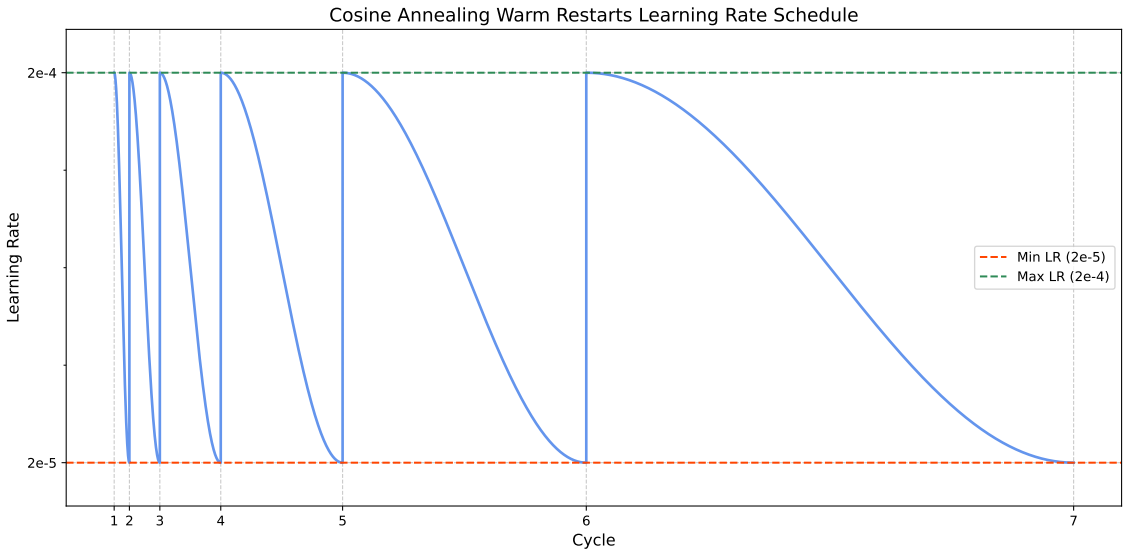

Figure 2.1: Cosine Annealing Warm Restarts Learning Rte Schedule

Figure 2.1 illustrates the Cosine Annealing Warm Restarts learning rate schedule, an adaptive learning rate policy. The x-axis represents training steps, while the y-axis

shows the learning rate. The schedule oscillates between a minimum learning rate of 2e−5 and a maximum of 2e−4. Each cycle follows a cosine curve, starting at the maximum rate and gradually decreasing to the minimum before abruptly resetting. The distinctive feature of this schedule is the increasing cycle length over time, with each new cycle being twice as long as the previous one. This approach combines the benefits of cosine annealing, which allows for smooth transitions between learning rates, with periodic restart that reset the rate to its maximum value. The gradual extension of cycle length aims to provide more fine-tuning time in later stages of training while still maintaining the ability to escape local optima.

2.5.2 Cyclical Learning Rate schedulers

In the family of Cyclical Learning Rate schedulers [37], one common approach is to use a triangular2 policy. This method oscillates the learning rate between predefined lower and upper bounds. Notably, after each cycle the maximum learning rate is reduced by a factor of two. This adaptive oscillation ensures that the early stages of training are dominated by exploration, while later stages emphasize stability and fine-tuning.

The following pseudocode (Algorithm 4) provides a detailed implementation of the Triangular2 policy learning rate scheduling algorithm:

Algorithm 4 Cyclical Learning Rate with Triangular2 Mode

Require: Base learning rate lr_{base}, maximum learning rate lr_{max}, up-step size $step_{\text{up}}$, down-step size $step_{\text{down}}$, total number of iterations N

$iteration \leftarrow 0$

$TotalStepsPerCycle \leftarrow step_{\text{up}} + step_{\text{down}}$

for $t \leftarrow 1$ to N **do**

Compute current cycle:

$$cycle \leftarrow \left\lfloor 1 + \frac{iteration}{TotalStepsPerCycle} \right\rfloor$$

Compute relative position within cycle:

$$x \leftarrow \left| \frac{iteration \bmod TotalStepsPerCycle}{step_{\text{up}}} - 1 \right|$$

Adjust the maximum learning rate for the current cycle:

$$lr_{\text{max, current}} \leftarrow lr_{\text{max}} \cdot 2^{-(cycle-1)}$$

Compute current learning rate:

$$lr \leftarrow lr_{\text{base}} + (lr_{\text{max, current}} - lr_{\text{base}}) \cdot \max(0, 1 - x)$$

Update optimizer with lr

$iteration \leftarrow iteration + 1$

end for

Ensure: Final learning rate schedule applied.

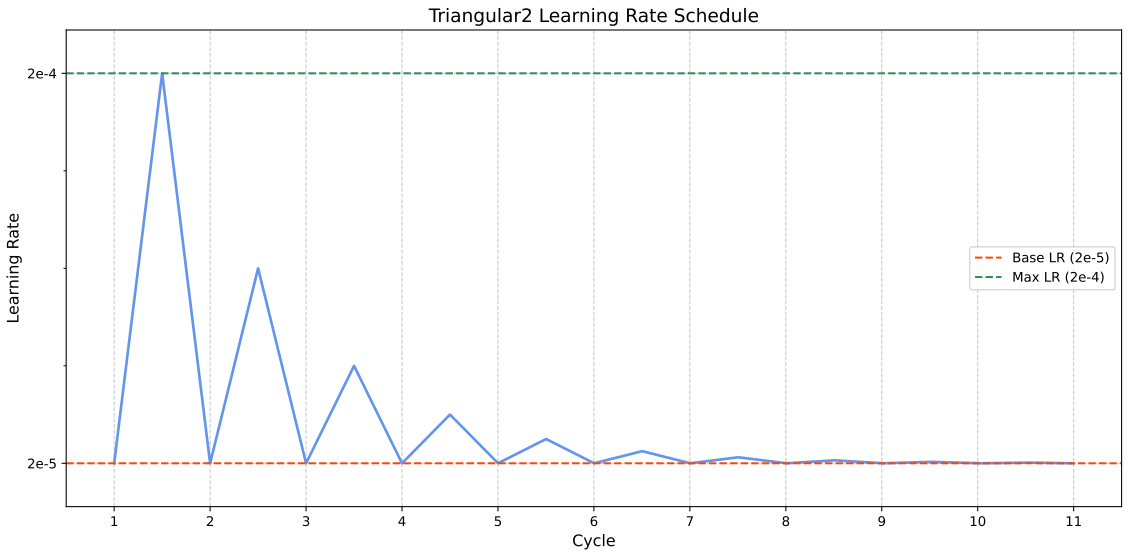

Figure 2.2: Triangular2 Learning Rate Schedule

Figure 2.2 demonstrates the Triangular2 learning rate schedule mode, a cyclical learning rate mode. The x-axis represents training steps, while the y-axis shows the learning rate. The schedule oscillates between a base learning rate of 2e−5 and a maximum of 2e−4. Each cycle comprises two phases: a linear increase from the base to the maximum rate, followed by a linear decrease back to the base rate. The distinctive feature of Triangular2 is the gradual reduction in the cycle amplitude over time, which is evident in the decreasing peaks of subsequent cycles. This adaptive approach aims to balance exploration of the parameter space with fine-tuning as training progresses, potentially leading to improved model convergence and performance.

Both scheduling strategies are designed to enhance convergence efficiency and promote improved generalization. While the cosine annealing approach periodically facilitates exploration through restarting the learning rate, the cyclical method with the Triangular2 policy systematically attenuates the amplitude of learning rate oscillations, thereby integrating the benefits of both exploratory and exploitative phases. A detailed comparative analysis of these strategies is provided in the experimental section.

3

Methods

In this section, we present the architectures of our hybrid CNN-GNN models, detail the dataset construction process, outline the training methodology, and describe the metrics employed for evaluation.

3.1 Model Architectures

This section outlines the architecture of the proposed model, which captures protein-ligand interactions using a graph neural network framework. The model processes structural input through a series of modular components, each responsible for transforming and exchanging information within and between graphs. Figure **??** illustrates the overall design and data flow.

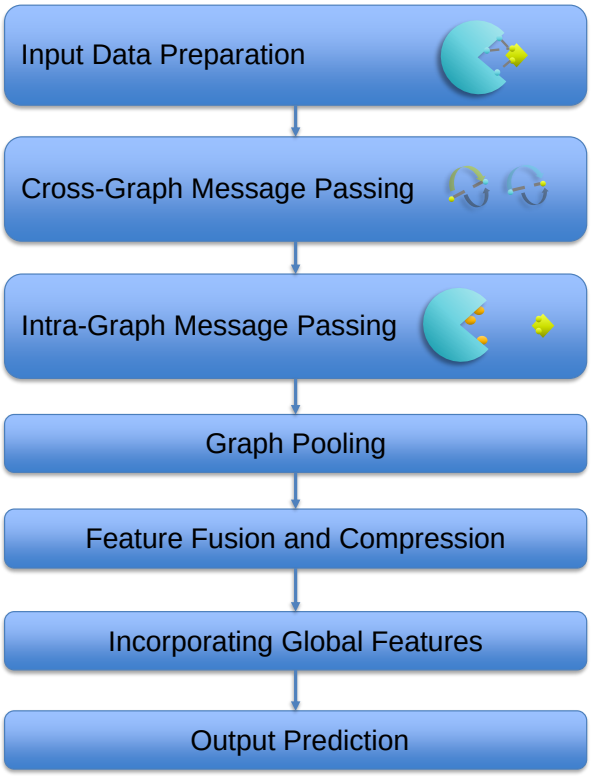

Figure 3.1: Model architecture of the Graph Neural Networks for protein-ligand binding affinity prediction.

3.1.1 Input Data Preparation

The input is represented as two separate graphs, one for the protein and one for the ligand. In both graphs, nodes correspond to atoms, and edges reflect either covalent bonds or spatial proximity. For the protein, only selected atoms (e.g., backbone or heavy atoms) are included to reduce complexity. In addition to intra-graph connections, cross-graph edges are established between protein and ligand atoms that are within a certain distance threshold, allowing the model to capture potential interaction sites across the two structures.

3.1.2 Cross-Graph Message Passing

A key component of the model is the message passing between the protein and ligand graphs via the cross-graph edges. This allows information to flow in both directions, enabling the model to capture how atoms from one molecule influence those in the other—something not possible by treating each graph independently.

3.1.3 Intra-Graph Message Passing

Following the cross-graph updates, message passing is applied within each graph separately. In this stage, nodes aggregate features from their neighbors to capture

local structural context. This process can utilize standard GNN layers with linear transformations, GAT that dynamically weigh neighbor importance, or STA methods that model higher-order interactions. By switching between linear layers, GAT, or STA in this component, the model can be adapted to balance computational efficiency and representational power based on the specific needs of the task.

3.1.4 Graph Pooling

To obtain fixed-size representations for the protein and ligand, node embeddings are pooled across each graph. We use attention-based methods to summarize atomic information into compact graph-level feature vectors.

3.1.5 Feature Fusion and Compression

The protein and ligand graph features are concatenated into a single vector, which is then passed through dimensionality reduction layers. These layers help focus on the most relevant features while keeping the representation compact, improving generalization.

3.1.6 Incorporating Global Features

If available, additional descriptors—such as molecular fingerprints or other domain-specific features—can be included at this stage. These global features are fused with the learned graph representation, allowing the model to leverage both data-driven patterns and prior knowledge.

3.1.7 Output Prediction

The final representation is passed to a prediction layer, which outputs a value such as binding affinity or a classification label. The model is trained end-to-end using appropriate loss functions and optimized using standard gradient-based methods.

3.2 Dataset

In this project, we utilize a publicly available dataset known as the CrossDocked2020 dataset, which is an augmented version derived from the original PDBBind dataset. In this section, we will first introduce the PDBBind dataset, then elaborate on the methods employed by CrossDocked2020 to enhance and expand the data based on PDBBind. Finally, we will outline how the CrossDocked2020 dataset is specifically utilized within our project.

3.2.1 Dataset

PDBBind Data Set The **PDBbind database**, established in 2004 by Shaomeng Wang and collaborators [38], provides a curated compilation of experimentally derived binding affinity measurements for biomolecular complexes documented in the Protein

Data Bank (PDB) [39]. Binding affinities within this dataset are quantitatively represented as pK values ($-\log K_d$, $-\log K_i$, or $-\log IC_{50}$), with a significant portion (approximately 39%) falling between the values of six and eight, indicating a balanced coverage across affinity ranges.

For each complex, three-dimensional structures of the receptor and its ligand-binding site, specifically defined by residues positioned within a 10 Å radius around the ligand, are accessible as standard-format PDB files (`.pdb`). Ligands are provided separately in two chemically informative file formats—MOL2 and SDF. MOL2 files encompass comprehensive molecular data, including atomic coordinates, bond structures, charges, and electronic states, essential for computational modeling tasks. The SDF format, an expanded form of the MOL file, contains multiple molecular entries along with supplementary chemical metadata. Ligand structures in these formats may either be sourced directly from chemical databases (e.g., PubChem or DrugBank) or converted from PDB format using software tools such as OpenBabel or RDKit.

The database is structured into three progressively selective subsets: (1) the comprehensive **general set**, which includes all protein–ligand complexes; (2) the carefully screened **refined set**, extracted from the general set based on stricter quality benchmarks; and (3) the representative **core set**, which is derived from the refined subset to serve as a benchmark for computational model evaluation. Criteria defining the refined set include crystal structures resolved at 2.5 Å resolution or better, complexes exclusively stabilized by non-covalent interactions, experimental verification of binding affinity through K_i or K_d measurements, consistency between crystallized and assay-tested protein–ligand pairs, ligands consisting exclusively of standard atoms with molecular weights not exceeding 1000 Da, and absence of unnatural amino acids in protein binding sites. Formation of the core set involves clustering refined structures based on protein sequence similarity with a threshold of 90%, requiring a minimum cluster size that varied across database releases (e.g., four complexes per cluster in 2010 while five in 2011).

Professor Renxiao Wang's research group manages and annually updates the PDBbind database. Originally limited to protein–small ligand interactions, the database broadened its scope in 2008 to include complexes involving protein–protein, protein–nucleic acid, and nucleic acid–ligand interactions. The latest significant update, PDBbind-2021, introduced expanded binding data, improved structure curation, additional analytical tools, and a new cloud-based platform. Additionally, version 2006 specifically corrected inaccuracies identified in the earlier 2005 release. The steady increase in Protein Data Bank entries ensures continual enrichment and growth of the PDBbind database, substantially aiding the development and rigorous evaluation of computational methodologies, particularly in deep learning and molecular modeling domains.

CrossDocked2020 Data Set The CrossDocked2020 data set was systematically created using structural information from the Protein Data Bank (PDB), curated and grouped according to ligand binding site similarities by Pocketome (version 17.12) [40]. The initial preparation involved downloading relevant protein structures

and applying rigorous preprocessing steps. Ligands exceeding a molecular weight of 1000 Da were excluded, while structural water molecules were removed. The cleaned protein structures were then aligned precisely to the binding site definitions provided by Pocketome using ProDy software [41]. Unlike standard practice with PDBbind datasets, ions detected by ProDy remained part of the receptor atoms in the final dataset.

Subsequently, a comprehensive docking procedure was performed in which each ligand associated with a specific pocket was systematically docked into all other receptor structures within the same pocket group using smina docking software [42]. This strategy significantly increased the number of docking poses. Binding affinities (represented by pK values) were sourced from the 2017 PDBbind database. An essential assumption during data labeling was that each ligand retained consistent affinity across receptors within the same pocket cluster, thereby permitting the use of aligned crystal structures as benchmarks for pose accuracy evaluation. Nevertheless, these simplifications inherently introduced labeling inaccuracies.

In contrast to datasets derived directly from PDBbind, CrossDocked2020 utilized the predefined interaction clusters provided by Pocketome. This clustering facilitated internal comparisons among protein-ligand complexes that share the same binding pocket. The ProBiS algorithm [43], with a selected z-score cutoff of 3.5, was employed to assemble pockets into clusters that were randomly assigned across cross-validation folds.

To enhance the robustness and generalization capabilities of models trained on the CrossDocked2020 dataset, an iterative data refinement procedure was implemented. Initially, a baseline model was trained using the complete set of docking poses. This trained model was then leveraged to reassess and rank these poses, highlighting particularly challenging instances based on discrepancies between predicted and true structural alignments. Specifically, inactive examples were selected according to two main criteria: highly scored poses (above 0.9) that deviated notably from their corresponding crystal structure (greater than 2 Å RMSD), or poorly scored poses (below 0.5) that closely resembled their respective crystal structures (less than 2 Å RMSD). Each added inactive example was ensured to be distinct from existing dataset entries by enforcing a minimum RMSD difference of 0.25 Å. This iterative cycle was executed twice, each successive iteration yielding progressively fewer additional poses. Ultimately, due to diminishing returns observed after the second iteration, further iterations were deemed unnecessary considering the associated computational costs.

In total, the dataset encompasses 13,780 distinct ligands, with approximately 41.9% accompanied by known binding affinity data, distributed across 2,922 unique binding pockets. These were organized into 18,450 protein-ligand complexes, collectively generating around 22.6 million docking poses.

3.2.2 Data Preprocessing

Data Downsampling
The original dataset of CrossDocked2020 exhibits significant class imbalance, as

summarized in Tables 3.1 and 3.2:

Table 3.1: Overall Distribution of Labels

Label	Count	Percentage (%)
Inactive	21,740,871	96.34
Active	825,577	3.66

Table 3.2: Distribution of pK values across labels

pK Category	Inactive	Active	Percentage of Total (%)
$pK < 0$	8,913,284	0	39.50
$pK = 0$	12,827,587	473,666	58.94
$pK > 0$	0	351,911	1.56

In this project, we do not utilize the entire dataset for model training; instead, we apply a downsampling strategy to handle the imbalance in the dataset. Our downsampling procedure is detailed as follows: Firstly, we select all active poses from the original dataset with a pK value greater than 0, yielding a total of 351,911 active poses. Conversely, inactive poses, characterized by pK values less than 0, amount to 8,913,284. Given the significant numerical disparity between inactive and active poses, we apply downsampling to the inactive poses to balance the dataset. The sampling strategy involves selecting inactive poses that correspond directly to the ligands associated with active poses. To illustrate, if the binding pocket labeled `1433S_HUMAN_1_233_0` contains 9 active poses corresponding to the ligand `3t0m_A_rec_4dhp_y07_lig`, we sample an equivalent number of inactive poses from the same binding pocket with the same name as `3t0m_A_rec_4dhp_y07_lig`. Subsequently, we partition the active and inactive poses within each binding pocket into training, validation, and test sets using a ratio of 0.7, 0.15, and 0.15, respectively. This ensures that each subset maintains equal proportions of active and inactive poses, and that the distribution of binding pockets remains consistent across training, validation, and test datasets, as well as the entirely downsampled dataset. Following this procedure, we obtained a balanced subset of data suitable for model training, as summarized in Table 3.3,and the overall distribution of pK values is shown in Figure 3.2.

Table 3.3: Summary of DownSampled Data for Model Training

Dataset	Counts
Entire Data	700,471
Active (pK > 0)	351,911
Inactive (pK < 0)	348,560
pK distribution in entire data	
pK $\in (4, 15)$	326,638
pK $\in (0, 4)$	25,273
pK $\in (-4, 0$	24,158
pK $\in (-15, -4)$	324,402
Train	643,100
Validation	27,214
Test	30,157

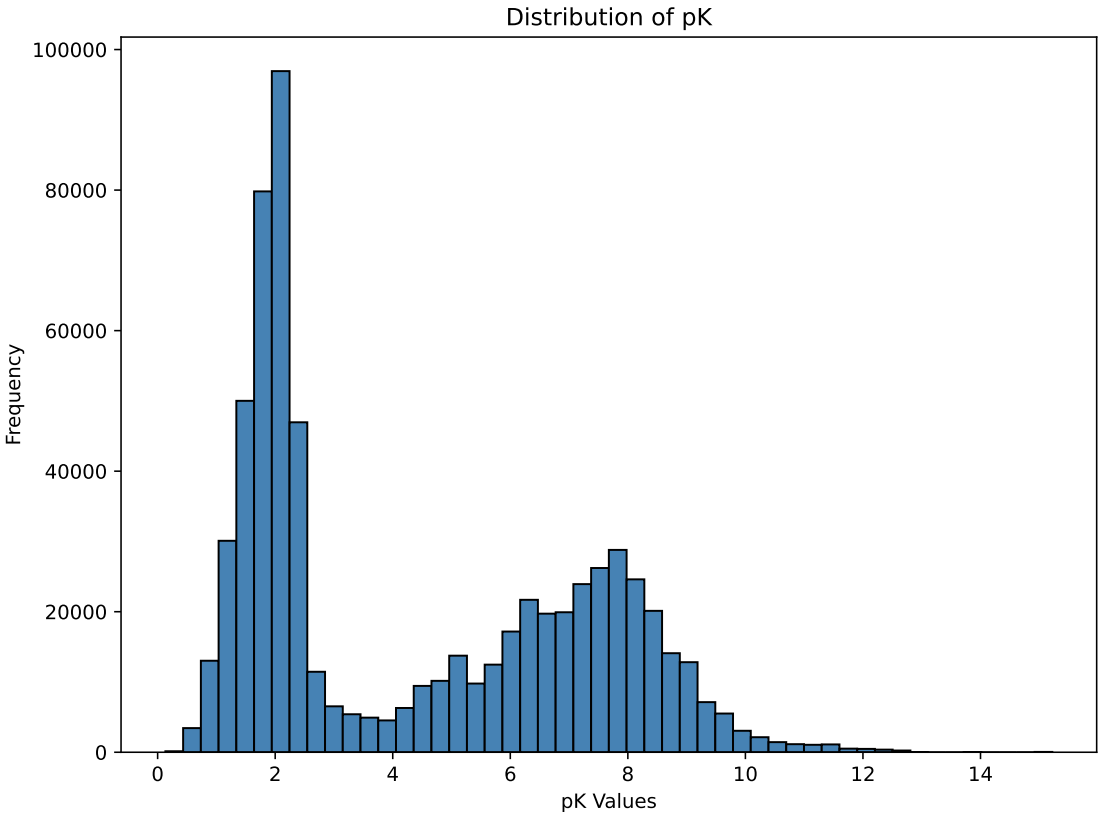

Figure 3.2: Overall distribution of pK values after downsampling.

Data Preprocessing

Our data preprocessing pipeline transforms raw molecular data into a format optimized for machine learning analysis, focusing specifically on the critical features of protein–ligand interactions. The process is organized into several key stages:

Molecular Structure Preparation: Protein structures are initially truncated to emphasize the binding site, reducing computational overhead while preserving essential interaction details. Ligand structures are prepared simultaneously, with careful attention paid to accurate atom typing and bond order assignment.

Graph Representation: We convert both protein and ligand structures into graph representations, where atoms serve as nodes and chemical bonds form the edges. This conversion facilitates the capture of inherent topological and spatial relationships within molecules, setting the stage for graph-based machine learning techniques.

A comprehensive set of features is extracted at multiple levels:

- **Proteins:** Atom-level attributes such as element type, hybridization state, and local chemical environment are determined, along with residue-level details to provide a full biochemical context.
- **Ligands:** In addition to atom-level features, molecular fingerprints are computed to encapsulate overall chemical characteristics.
- **Edge Features:** Derived from bond properties and interatomic distances.

Data Integration and Normalization: The molecular graphs are integrated with interaction profiles and supplementary experimental or computational properties (including binding affinities and RMSD values). Numerical features are normalized to ensure consistent scaling across the dataset.

This comprehensive approach, combining structural information with detailed interaction profiles, provides a robust foundation for subsequent modeling and analysis of molecular binding phenomena.

3.3 Training Procedure

3.3.1 Training Configuration and Continuation Strategy

The training process initiates with a randomized shuffling of the training samples to mitigate ordering biases. Each mini-batch consists of 128 samples. Considering the dataset is approximately balanced across target labels (e.g., active vs. inactive samples), no stratified sampling is employed. The optimization is conducted using Adam with a base learning rate of 2×10^{-5}, managed by a cyclic learning rate scheduler to adaptively modulate the learning rate throughout training. Continuation training is fully supported: prior checkpoints including model parameters, optimizer states, and validation loss records are automatically reloaded if available. Additionally, the previously best-performing model, identified by the lowest validation loss, is restored to ensure continuity in performance tracking.

3.3.2 Dynamic Dataset Management and Loader Scheduling

The substantial scale and heterogeneity of the training dataset pose significant challenges in both memory consumption and computational efficiency. Directly

loading all data into memory during each epoch would not only risk memory overflow but also considerably prolong the training time. To address these constraints, we partition the full dataset—including primary sources such as *CrossDocked2020* and auxiliary datasets—into smaller, more manageable chunks, where each chunk corresponds to a specific subset of the data and is handled by an independent data loader.

All data loaders, regardless of their source domain, are systematically organized into a unified queue structure. Rather than activating all loaders simultaneously, only a dynamically selected subset is utilized during each training epoch. Over successive epochs, this subset evolves, ensuring that the model is gradually exposed to the entirety of all available datasets without exceeding memory limitations. For instance, if the collective dataset is divided into twenty chunks across multiple sources, a typical epoch may involve training on only two or three of them, with the active subset rotated periodically to guarantee comprehensive coverage over time.

To further enhance learning diversity and computational efficiency, the number of active loaders per epoch is not static. During the early stages of training, a larger number of loaders is employed to maximize exposure to heterogeneous data distributions across different domains. As training progresses, the subset size is gradually reduced, allowing the model to focus on fine-tuning representations learned earlier. This progressive narrowing of the learning scope facilitates faster convergence while preserving the benefits of broad initial exposure.

Loader selection follows a round-robin and interleaved scheduling strategy, as shown in Section 2.4.1 and 2.4.2. After each epoch, the loader queue is rotated such that overlapping subsets are formed across consecutive epochs. For example, given loaders A, B, and C, the subset selection may proceed as AB $\rightarrow$ BC $\rightarrow$ CA $\rightarrow$ AB, and so forth. This controlled overlap between epochs promotes smoother transitions between data domains, encourages knowledge sharing, and mitigates abrupt distributional shifts that could destabilize learning. In this project, we split the full training dataset into 20 equally sized chunk files, each representing 5% of the total training data. The data distribution within each chunk—including binding pockets, complexes, and their active/inactive labels—is kept consistent with that of the original, unchunked dataset. Therefore, the subset of the training set per epoch will be chunk 0 and chunk 1 $\rightarrow$ chunk 1 and chunk 2 $\rightarrow$ chunk 2 and chunk 3 $\rightarrow$ chunk 3 and chunk 4, and so on if only 2 chunk files, i.e., 10% of the training data, will be employed in each epoch in the training process.

Compared to traditional round-robin or naive interleaved training schemes, our customized scheduling strategy introduces two critical improvements: (i) dynamic adjustment of subset sizes aligned with training progression, and (ii) explicit control over inter-epoch overlap across multiple datasets. These enhancements not only enable full utilization of the diverse training corpus without overwhelming system resources, but also foster cross-domain knowledge transfer, reduce the risk of overfitting to narrow data slices, and mitigate catastrophic forgetting when switching between sources. Ultimately, this approach leads to more stable training dynamics and improved generalization performance across heterogeneous tasks.

3.3.3 Epoch-wise Training Flow

During each training epoch, the model is sequentially optimized across the selected subset of data loaders. For every batch, the loss value is recorded, and predictions are accumulated. After completing training across all batches and loaders for the current epoch, an aggregated training loss is computed. Simultaneously, predicted and ground-truth labels are consolidated and saved into comma-separated value (CSV) files for subsequent analysis. This per-epoch aggregation of losses and outputs provides a structured foundation for monitoring training dynamics over time.

3.3.4 Validation and Performance Metrics

Upon completing the training phase within each epoch, the model undergoes validation using a designated set of validation loaders. Additional validation sources can be included if available. During validation, predictions are collected and compared against true labels, and key evaluation metrics—including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2)—are computed. These metrics, along with the validation loss, are systematically logged. This comprehensive validation framework facilitates continuous assessment of model generalization capabilities throughout the training process.

3.3.5 Periodic Testing on Held-out Data

To systematically assess generalization performance, the model is periodically evaluated on an independent held-out test set, typically every ten epochs. During these evaluations, predictions are compared against experimental labels, and the results are archived in CSV format. Furthermore, correlation-based measures, namely Pearson's correlation coefficient (R) and Kendall's Tau (τ), are calculated to capture both linear and ordinal relationships between predictions and ground truths. Periodic testing provides early indicators of overfitting or divergence, allowing corrective actions to be considered if necessary.

3.3.6 Checkpointing and Best Model Preservation

To safeguard against unexpected interruptions and facilitate recovery, model checkpoints—including model weights and optimizer states—are regularly saved at predefined intervals. Concurrently, if a model achieves a new minimum in validation loss, it is preserved separately as the current best model. This dual-saving strategy ensures both continuity of training and retention of the highest-performing model instance for subsequent deployment or analysis.

3.3.7 Early Stopping and Convergence Monitoring

An early stopping mechanism is incorporated to optimize computational efficiency and prevent overfitting once model performance stagnation is detected. After the model has been exposed to all training data loaders, its performance on the held-out

test set is monitored using Pearson's R and Kendall's τ. If neither metric exhibits a substantial improvement (e.g., greater than 0.01 increment) over a defined patience window, training is terminated. This dual-metric convergence criterion balances the need for sufficient exploration with early termination upon convergence, contributing to both training stability and computational efficiency.

3.3.8 Visualization and Result Analysis

At specified intervals, typically after every ten epochs, the prediction results are visualized by plotting predicted values against experimental labels. These plots are systematically saved for post-hoc analysis. Such visualizations serve as intuitive complements to quantitative metrics, enabling the identification of systematic biases, distributional anomalies, or potential outliers.

3.4 Evaluation Metrics

3.4.1 Overview

To comprehensively assess model performance in predicting binding affinities, we employed a combination of error-based and correlation-based evaluation metrics. Specifically, we evaluate the model using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error (MSE), the coefficient of determination (R^2), Pearson's correlation coefficient (R), and Kendall's Tau (τ). Each metric captures distinct aspects of model behavior, enabling a thorough and multi-faceted evaluation of both accuracy and consistency.

3.4.2 Root Mean Squared Error (RMSE)

RMSE quantifies the average magnitude of prediction errors, emphasizing larger deviations more heavily due to its quadratic nature. It is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{3.1}$$

where y_i denotes the experimental binding affinity, $\hat{y}_i$ represents the predicted value, and N is the total number of samples. A lower RMSE indicates greater overall prediction accuracy, particularly penalizing substantial outliers.

3.4.3 Mean Absolute Error (MAE)

MAE measures the average absolute deviation between predicted and true values. Unlike RMSE, it treats all deviations equally without disproportionately amplifying larger errors. MAE is defined as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.2}$$

Lower MAE values indicate higher predictive reliability across the dataset, especially in scenarios where balanced error treatment is preferred.

3.4.4 Mean Squared Error (MSE)

MSE provides another perspective on error magnitude by averaging the squared prediction errors:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.3}$$

MSE is closely related to RMSE (since RMSE is simply the square root of MSE) and emphasizes larger errors, making it useful for detecting variance in prediction inaccuracies.

3.4.5 Coefficient of Determination (R^2)

The coefficient of determination (R^2) measures the proportion of variance in the experimental outcomes that is predictable from the model's predictions. It is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{3.4}$$

where $\bar{y}$ represents the mean of the true values. An R^2 value closer to 1 indicates that the model explains a large portion of the variability in the experimental data, whereas an R^2 closer to 0 suggests poor explanatory power.

3.4.6 Pearson's Correlation Coefficient (R)

Pearson's R quantifies the strength of linear correlation between predicted and experimental values. It is computed as:

$$R = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{3.5}$$

where $\bar{y}$ and $\bar{\hat{y}}$ denote the means of the true and predicted values, respectively. Values of R closer to 1 signify stronger positive linear relationships, indicating that the model preserves the scale and directionality of affinity variations.

3.4.7 Kendall's Tau (τ)

Kendall's Tau (τ) assesses the ordinal association between predicted and experimental values by measuring the consistency in ranking order. It is defined as:

$$\tau = \frac{C - D}{\frac{1}{2}N(N-1)} \tag{3.6}$$

where C and D represent the number of concordant and discordant pairs, respectively. Higher values of τ imply better agreement in the relative ranking of predicted versus true affinities, independent of their absolute magnitudes.

3.4.8 Comprehensive Assessment and Interpretation

Together, these metrics provide a robust framework for performance evaluation. RMSE, MAE, and MSE quantify the absolute prediction errors, each with different sensitivities to error magnitude and outliers. The coefficient of determination (R^2) measures how well the model explains the variance in the data. Pearson's R evaluates the strength of linear correlation, while Kendall's τ measures the consistency in prediction rankings. Lower RMSE, MAE, and MSE values, combined with higher R^2, R, and τ scores, collectively indicate a model that is both accurate and generalizable across different affinity prediction tasks.

4

Results

In this section, we present a comprehensive evaluation of our protein-ligand affinity model, systematically exploring various factors that influence its predictive performance. Building upon the foundational analysis of message-passing schemes (including GAT, STA, and linear baseline models), we first investigate the impact of different learning rate strategies. This includes static learning rate, adaptive schedules, each designed to optimize training stability and convergence while minimizing overfitting. These strategies were implemented to assess the upper performance limits of our model and identify the most effective learning configurations for this challenging biochemical prediction task.

To further evaluate the model's generalization capacity, we adopted a clustered cross-validation (CCV) framework. This approach, which groups protein-ligand pairs based on structural similarity, provides a rigorous test of the model's ability to make accurate predictions on previously unseen complexes, effectively simulating real-world application scenarios. This is particularly important for graph-based architectures, where the risk of overfitting to specific molecular substructures can obscure true generalization capability.

In addition to these primary analyses, we conducted a detailed error breakdown, focusing on false positives (FP), false negatives (FN), and prediction errors. This component of the evaluation aims to uncover subtle performance variations across different prediction regimes, offering insights into the model's strengths, weaknesses, and potential biases. By examining the distribution of these error classes, we aim to identify patterns that could inform further model refinements or targeted training strategies.

4.1 Feasibility Validation of the Customized Training Process

Given the considerable size of our training dataset, conventional training pipelines often encounter significant challenges, including frequent out-of-memory (OOM) errors and prohibitively long training times. To address these constraints, we adopted a dynamic dataset management and dataloader scheduling strategy, as detailed in Section 3.3.2. This approach is designed to optimize memory utilization and streamline the training data loading process, thereby reducing computational

overhead and ensuring more efficient training.

However, before fully integrating this approach into our project, it is critical to validate its foundational reliability through a comprehensive sanity check. This initial step serves multiple essential purposes: it ensures that the new strategy provides the intended performance gains, verifies the absence of critical errors or data inconsistencies, and establishes a baseline for subsequent, more complex evaluations. Without such a preliminary assessment, the risk of introducing subtle yet potentially severe biases into the training process remains significant. Therefore, this sanity check is not merely a procedural formality, but a foundational safeguard to confirm that our optimized data handling strategy is both effective and robust under realistic training conditions.

To efficiently evaluate the feasibility of our dynamic training strategy, we first conduct a sanity check using a reduced data subset, i.e., 10% of the full training set per epoch. This approach is both pragmatic and resource-conscious: if the model is capable of achieving satisfactory predictive performance under this constrained data regime, it will justify subsequent scaling to larger data volumes during the formal training phase. Furthermore, the use of a smaller subset minimizes the computational burden, allowing for rapid iterative experimentation to identify potential flaws or unexpected behaviors in our training pipeline. This sanity check thus serves as a critical preliminary validation step before committing extensive computational resources to full-scale model training. Below are the Experimental Configurations, we trained the model for 300 epochs using 10 percent of the full training data, i.e., 2 chunk files, at each epoch, exploring multiple learning rates across three distinct architectures (GAT, STA, and Linear).

Model performance across different architectures
Table 4.1 summarizes the resulting model performances, which we evaluate using Pearson's correlation coefficient (R) and Kendall's Tau (τ) as evaluation metrics.

Table 4.1: Model performances (Pearson's R and Kendall's τ) on train and test sets.

Method	Train		Test	
	R	τ	R	τ
Linear	0.88	0.66	**0.72**	**0.52**
STA	0.86	0.65	0.72	0.51
GAT	0.89	0.66	**0.72**	**0.52**

Table 4.1 presents the Pearson correlation coefficient (R) and Kendall's Tau (τ) for three model architectures—Linear, STA, and GAT—evaluated on both the training and test sets. On the training set, GAT achieves the highest correlation ($R = 0.89$) and ties with Linear for the highest rank correlation ($\tau = 0.66$), indicating its superior capacity to fit both linear and ordinal relationships in the data. The Linear model closely follows ($R = 0.88$, $\tau = 0.66$), while STA attains slightly lower metrics ($R = 0.86$, $\tau = 0.65$). In contrast, all three architectures converge to an identical Pearson correlation of 0.72 on the test set, with Linear and GAT

both achieving the highest Kendall's Tau of 0.52 (compared to 0.51 for STA). This uniformity in test performance suggests that, when trained on only 10% of the full data, model complexity does not confer a clear generalization advantage. The generalization gap—measured as the decrease from train to test—amounts to approximately 0.16–0.17 in R and 0.14 in τ for Linear and GAT, and slightly less ($\Delta R \approx 0.14$) for STA. Such declines highlight a non-negligible overfitting tendency under extreme data scarcity, despite prolonged training (300 epochs). These findings carry two key implications. First, data volume is a primary constraint: under limited samples, simpler models (e.g., Linear) match or exceed the test performance of more expressive architectures. Second, to unlock the full potential of complex models like GAT, one must either augment the dataset or incorporate stronger regularization and early-stopping strategies.

Model performance across different learning rates

Figure 4.1, 4.2 and 4.3 show the training performance of three model architectures (Linear, STA, GAT) under various learning rates. Each subpanel depicts how Pearson's R and Kendall's τ change as the learning rate varies.

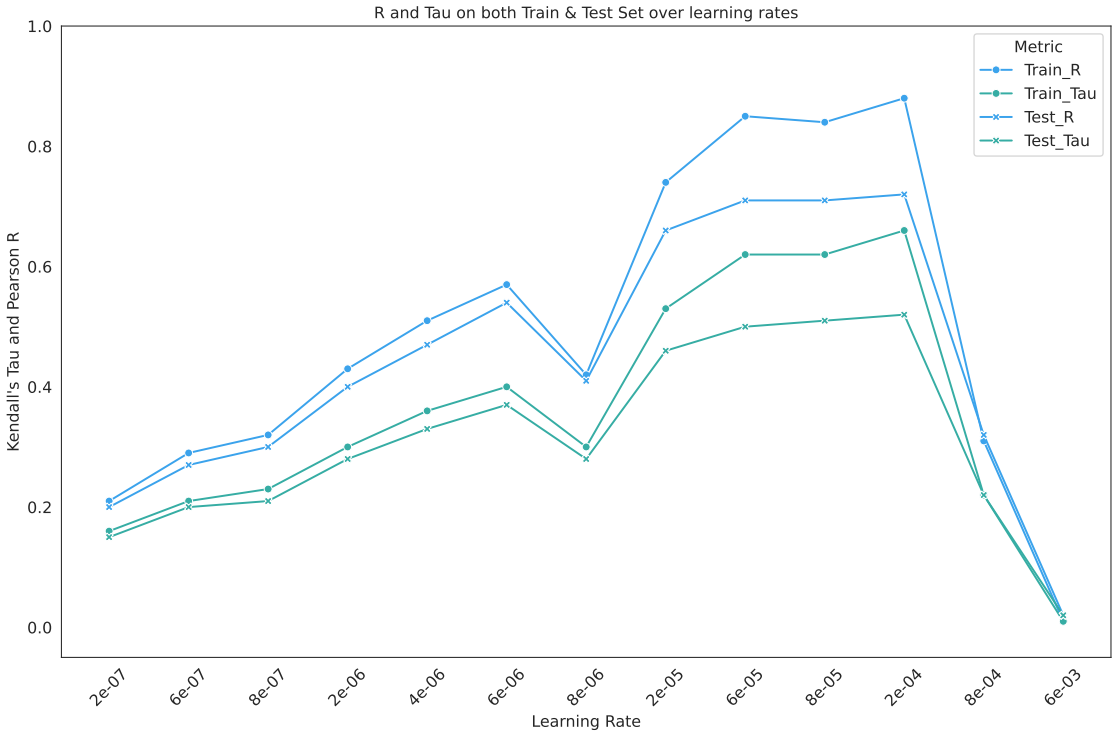

Figure 4.1: **Linear model** performance across various learning rates.

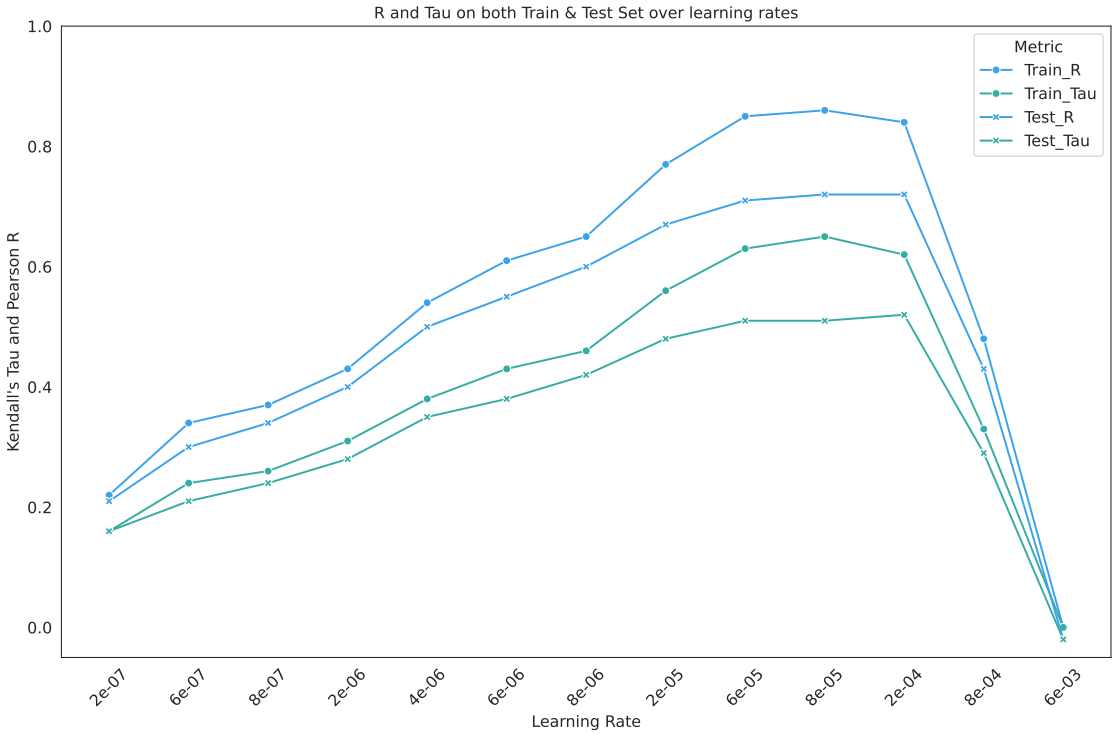

Figure 4.2: **STA model** performance across various learning rates.

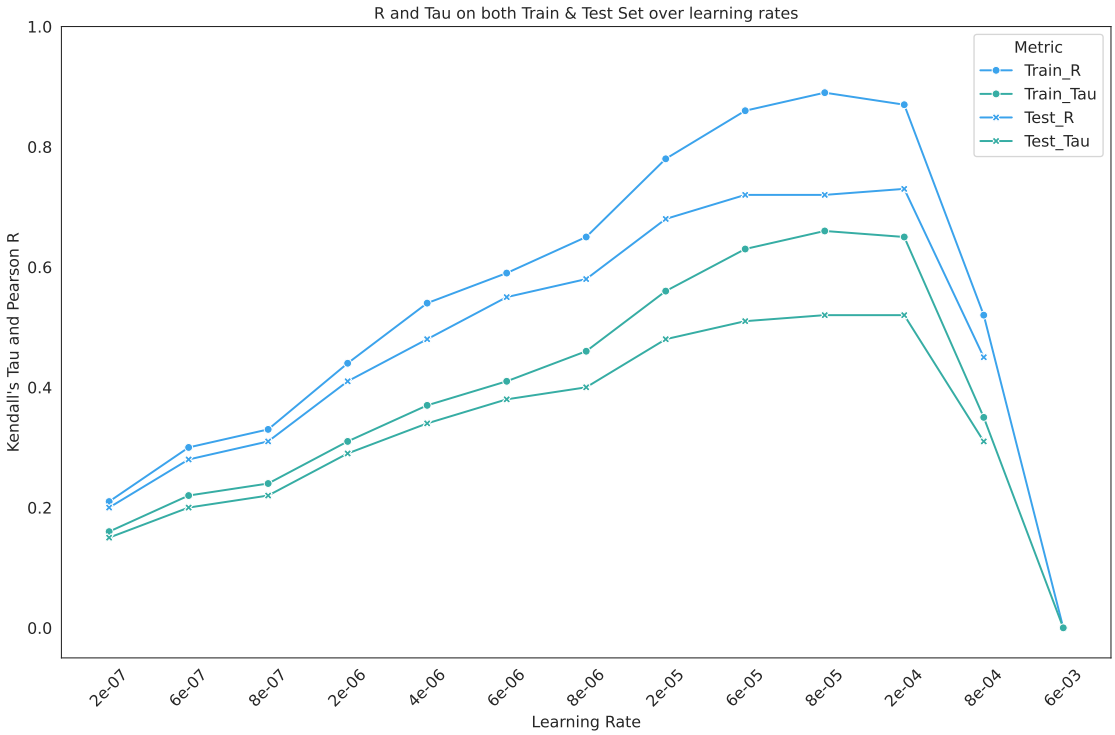

Figure 4.3: **GAT model** performance across various learning rates.

From Figures 4.1, 4.2, and 4.3, a detailed inspection reveals four distinct learning-rate

regimes that are consistent across Linear, STA, and GAT architectures. When the rate is very low ($\leq 1 \times 10^{-6}$), all models converge extremely slowly, with Pearson R and Kendall's τ lingering around 0.2–0.3 on both training and test sets, indicating under-fitting rather than over-fitting. In the moderate range from 1×10^{-6} to 2×10^{-5}, convergence accelerates markedly—for example, R rises from approximately 0.58 at 8×10^{-6} to about 0.68 at 2×10^{-5}—although a slight divergence between training and test curves signals the onset of mild over-fitting. Notably, while the GAT model exhibits the slowest start and the Linear model the fastest at low rates, all three architectures share inflection points in the 1×10^{-6} to 1×10^{-4} window, demonstrating that their sensitivity to the learning rate is broadly universal. The interval from 6×10^{-5} to 2×10^{-4} offers the best trade-off between convergence speed and generalization: training R peaks at 0.85–0.88 (with $\tau \approx 0.63$–0.66), and test R and τ reach 0.72–0.73 and 0.52, respectively, with diminishing marginal gains. Beyond 8×10^{-4}, performance degrades sharply, and at very high rates (for example, 6×10^{-3}) all metrics collapse to zero, indicating divergence due to overly aggressive updates. These findings motivate a fine-grained grid search within $[6 \times 10^{-5}, 2 \times 10^{-4}]$, the adoption of learning-rate schedulers such as warm-up with cosine decay or adaptive optimizers like AdamW, and the use of stabilization techniques such as gradient clipping or weight decay.

Training history across different architectures

Figure 4.4, 4.5, and 4.6 show the training history regarding the training loss and validation loss of three model architectures (Linear, STA, GAT) with the best learning rates.

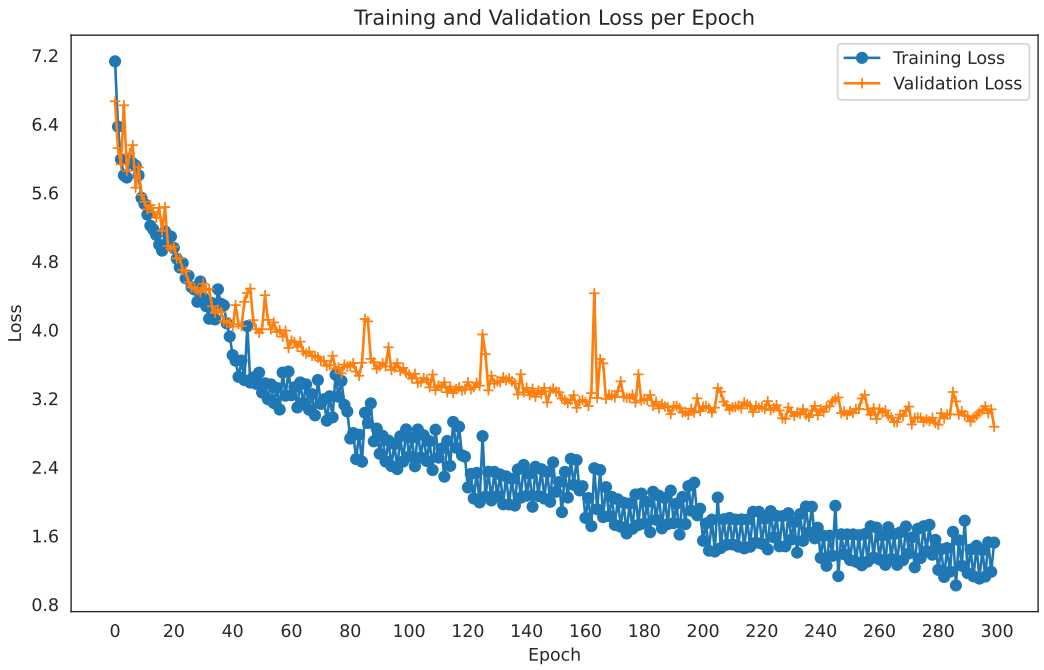

Figure 4.4: Training and validation loss per epoch for the **Linear model**.

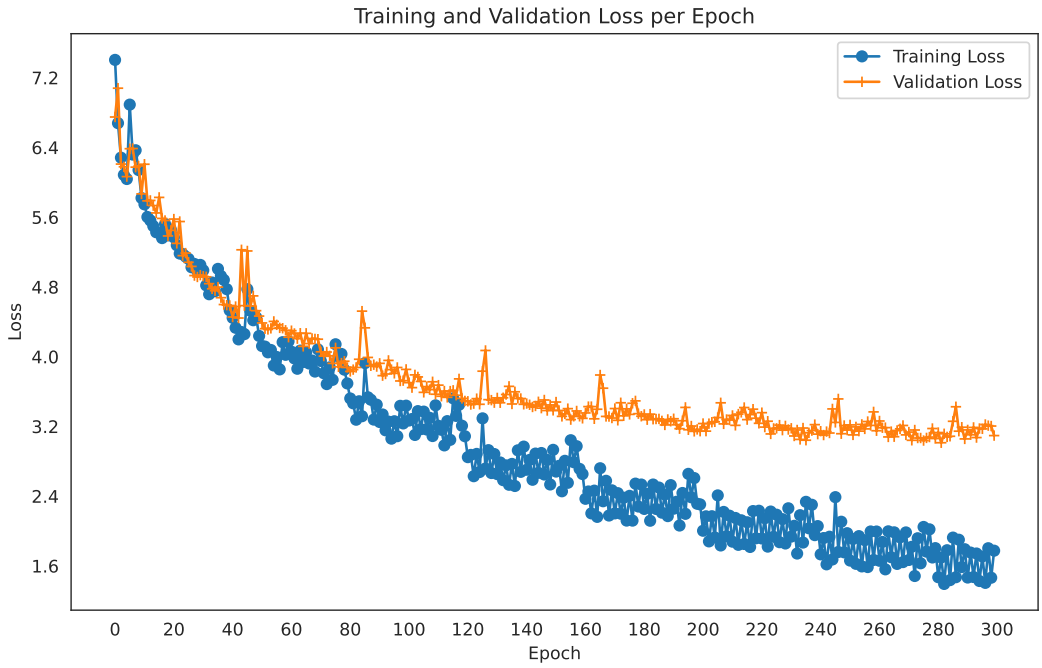

Figure 4.5: Training and validation loss per epoch for the **STA model**.

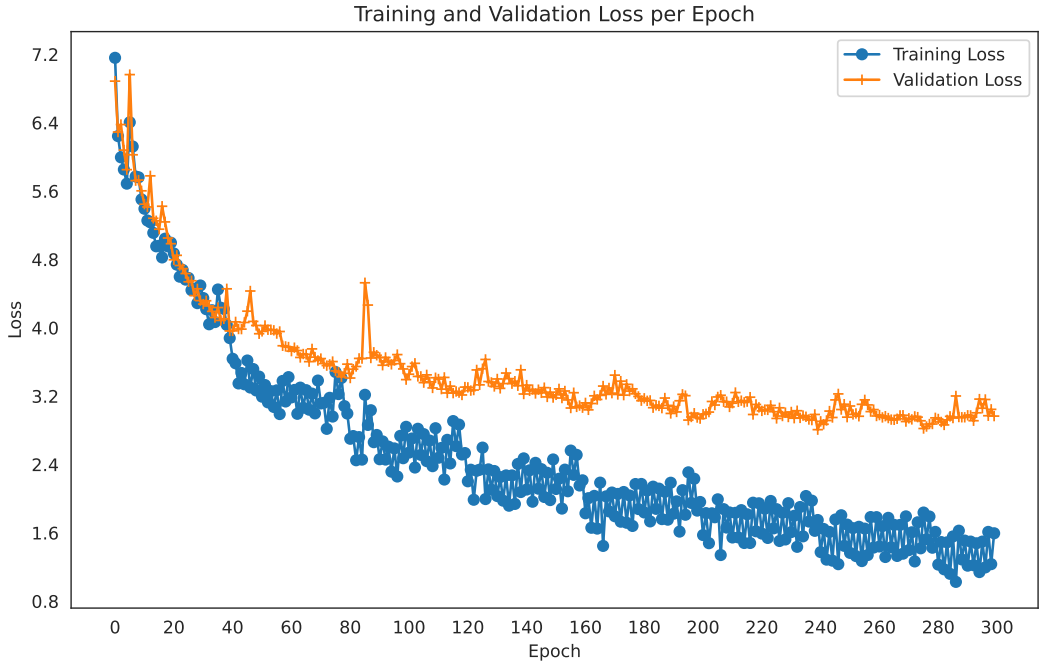

Figure 4.6: Training and validation loss per epoch for the **GAT model**.

A closer look at Figures 4.4, 4.5, and 4.6 reveals that all three architectures undergo

a two-phase learning process and exhibit a pronounced zigzag pattern in both training and validation loss curves. In the first 20–30 epochs, training loss drops precipitously—from above 7 down to approximately 4 for Linear and STA, and to around 4.5 for GAT—while validation loss follows a similar trajectory with slightly larger variance. Between epochs 30 and 80, the Linear model continues to reduce training loss rapidly, reaching near 2.5 by epoch 80, whereas STA plateaus a little higher at about 3.0, and GAT declines more slowly to around 3.5. Validation loss for Linear bottoms out around 3.2 at epoch 80 before slowly rising, indicating the onset of over-fitting; STA's validation loss reaches its minimum of roughly 3.4 at epoch 85 and then oscillates within a narrow band, suggesting stable generalization; GAT's validation loss only attains its lowest point near epoch 100 at around 3.6, accompanied by noticeable spikes that reflect sensitivity to its deeper attention mechanism. From epoch 80 onward, all models exhibit diminishing returns: training loss continues a gentle descent to final values of roughly 1.5 (Linear), 1.8 (STA), and 2.2 (GAT), while validation loss slowly drifts upward or oscillates around 3.1–3.5. The observed "zigzag" fluctuations arise because each epoch is trained on a largely new subset of data (only 5% overlap between consecutive epochs), yet ultimately the models still capture sufficient information. Among the three, Linear converges fastest but shows the largest train–validation gap, STA strikes the best balance between convergence speed and validation stability, and GAT—though ultimately capable of fitting complex patterns—requires more epochs and careful regularization to avoid noisy updates. These observations also suggest implementing early stopping around 80–100 epochs for Linear, around 100–120 epochs for STA and GAT, and incorporating learning-rate decay or dropout in the GAT architecture to further enhance generalization.

Experimental vs. predicted scatter on training and test sets across architectures.

Figure 4.7, 4.8, and 4.9 show the performance on the test set regarding the Pearson's correlation coefficient (R) and Kendall's Tau (τ) across three model architectures (Linear, STA, GAT) with the best learning rates.

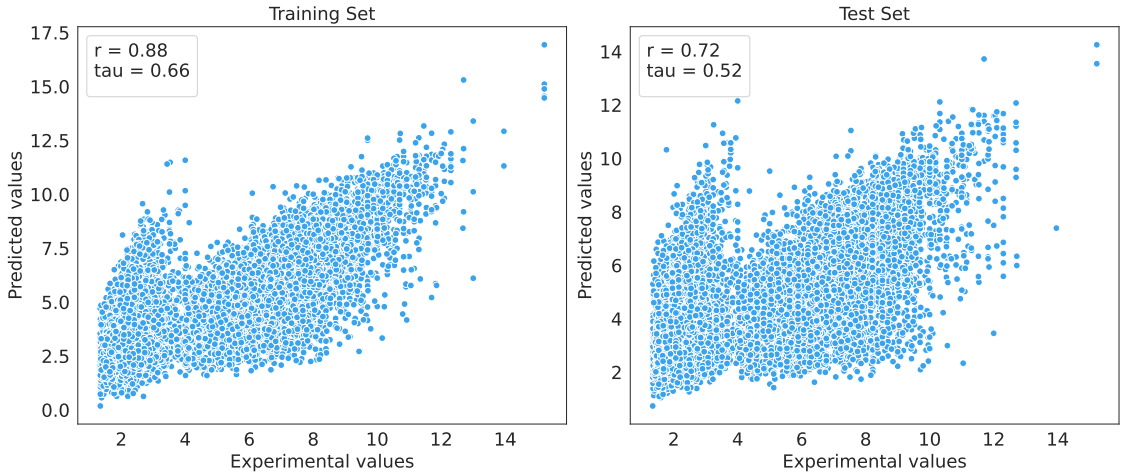

Figure 4.7: Experimental vs. predicted scatter on the train and test set for the **Linear model**.

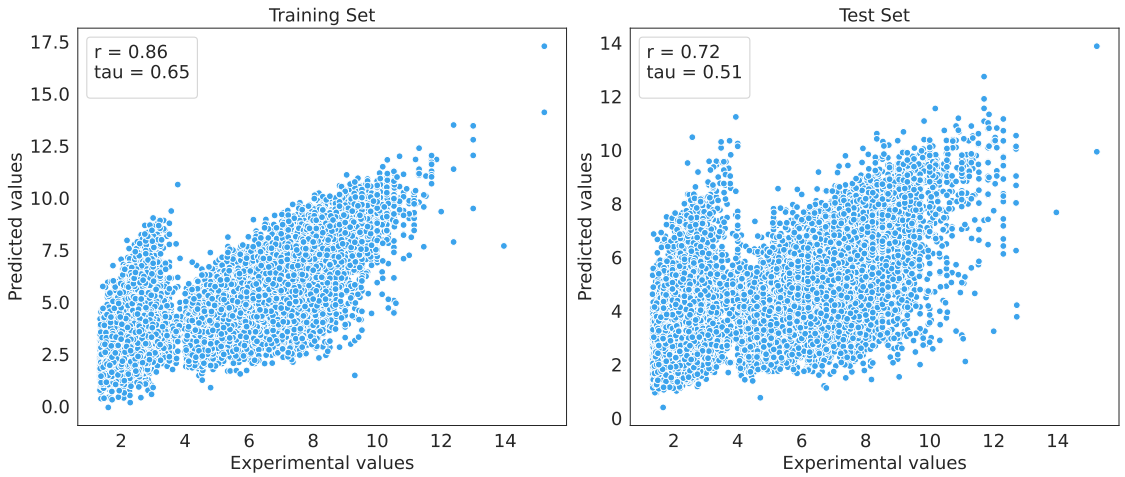

Figure 4.8: Experimental vs. predicted scatter on the train and test set for the **STA model**.

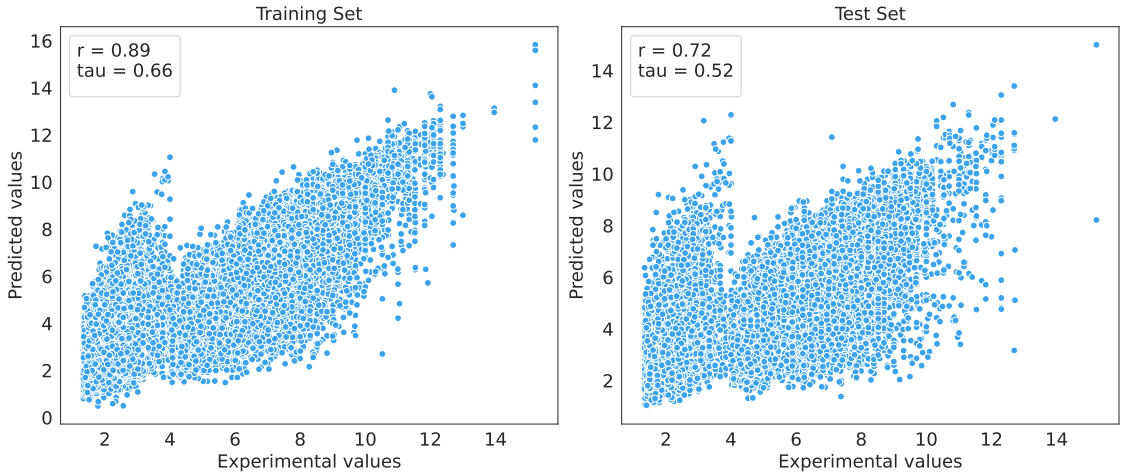

Figure 4.9: Experimental vs. predicted scatter on the train and test set for the **GAT model**.

Table 4.1 reports Pearson's R and Kendall's τ for the Linear, STA, and GAT architectures on both the training and test sets, while Figures 4.7, 4.8, and 4.9 present the corresponding scatter plots of experimental versus predicted values. On the training set, GAT attains the highest agreement ($R = 0.89$, $\tau = 0.66$), closely followed by the Linear model ($R = 0.88$, $\tau = 0.66$) and STA ($R = 0.86$, $\tau = 0.65$), indicating that all architectures capture both global trends and relative orderings with high fidelity. This is evidenced by the dense clustering of points along the identity line ($y = \hat{y}$) in the mid-range (4–10), whereas a few outliers at $x > 12$ exhibit slight underestimation. On the test set, performance degrades moderately—Linear and GAT both yield $R = 0.72$, $\tau = 0.52$, while STA achieves $R = 0.72$, $\tau = 0.51$—with a noticeably wider scatter around the ideal line and a downward shift of the mid-range density (4–8), reflecting increased variance on unseen data. The underestimation bias at extreme values persists and even amplifies (e.g. points near $x \approx 15$ predicted as $\hat{y} \approx 8$), suggesting difficulty in modeling rare observations. Despite similar aggregate metrics, the scatter plots reveal nuanced distinctions: the Linear baseline underfits the curvilinear trend, producing a tighter but flatter cloud at intermediate values; STA captures non-linear structure more effectively, reducing bias in the 6–10 range yet still underestimating the upper tail; and GAT achieves the best bias–variance trade-off, exhibiting the smallest vertical spread around the identity line for most of the distribution while likewise struggling at extremes. In summary, although all three models exhibit strong predictive performance across most of the data, they show specific weaknesses around values near 4—where accuracy degrades—and in forecasting extreme high-value observations, which are systematically underestimated.

In conclusion, after the initial sanity check, the models have demonstrated promising preliminary performance, even with only 10% of the full training data per epoch. As summarized in Table 4.1, the Linear model achieved a Pearson correlation coefficient (R) of 0.88 and a Kendall's Tau (τ) of 0.66 on the training set, with corresponding test set scores reaching **0.72** and **0.52**, respectively. The STA model showed comparable results, with $R = 0.86$ and $\tau = 0.65$ on the training set, and test set scores of **0.72**

and 0.51. Meanwhile, the GAT model exhibited the highest training set performance, reaching $R = 0.89$ and $\tau = 0.66$, along with test set scores of **0.72** and **0.52**, matching the Linear model's peak performance. Given these encouraging results, we plan to increase the per-epoch training data in the subsequent full-scale training phase, aiming to further enhance model generalization and predictive robustness.

4.2 Comprehensive Evaluation of Model's Performance

In this section, we first investigate the impact of different learning rate strategies on the performance of our protein-ligand affinity models. This process begins with a preliminary exploration using fixed learning rates to identify the optimal learning rate range that balances convergence speed and generalization ability while minimizing overfitting. Once this baseline range is established, we then employ various learning rate schedulers within this interval, including adaptive mechanisms designed to further refine the training dynamics. This two-stage approach allows us to systematically assess the upper performance limits of each model architecture, ultimately guiding the selection of the most effective training configurations for this challenging biochemical prediction task. All models in this section were trained for 300 epochs using 25% of the full training data, corresponding to 5 chunk files per epoch, to maintain a balance between computational efficiency and dataset diversity.

4.2.1 Selection of Fixed Learning Rates for Baseline Training

In this part of the experiments, we trained the model multiple learning rates. To identify an optimal fixed learning rate range, we systematically explored rates from 8×10^{-7} to 6×10^{-4} across three distinct architectures: GAT, STA, and Linear. This initial phase aimed to establish a strong performance baseline by selecting learning rates that balance convergence speed, stability, and generalization ability before applying more sophisticated learning rate scheduling strategies.

Model performance across different architectures
Table 4.2 summarizes the resulting model performances, which we evaluate using Pearson's correlation coefficient (R) and Kendall's Tau (τ) as evaluation metrics. Compared with the Section 4.1 of feasibility validation, shown in Table 4.1, increasing the per-epoch training data volume yields substantial improvements in both training and test performance. On the test set, the Linear model's Pearson correlation coefficient increases from $R = 0.72$ to $R = 0.75$ (absolute improvement of 0.03, relative improvement of approximately 4.2%), and Kendall's τ from $\tau = 0.52$ to $\tau = 0.55$ (absolute improvement of 0.03, relative improvement of approximately 5.8%); the GAT model achieves the best results, with R rising from 0.72 to 0.77 (absolute improvement of 0.05, relative improvement of approximately 6.9%) and τ from 0.52 to 0.56 (absolute improvement of 0.04, relative improvement of approximately 7.7%); STA remains inferior to both Linear and GAT. Consequently, our subsequent experiments will focus exclusively on the Linear and GAT architectures, and no further STA-based models will be trained.

Table 4.2: Model performances (Pearson's R and Kendall's τ) on train and test sets.

Method	Train		Test	
	R	τ	R	τ
Linear	0.94	0.74	**0.75**	**0.55**
STA	0.93	0.73	0.74	0.54
GAT	0.95	0.75	**0.77**	**0.56**

Model performance across different learning rates

Figure 4.10, 4.11 and 4.12 show the training performance of three model architectures (Linear, STA, GAT) under various learning rates. Each subpanel depicts how Pearson's R and Kendall's τ change as the learning rate varies. From these figures, it is evident that, after increasing the per-epoch training sample size, the model still converges very slowly at learning rates below 1×10^{-5}, while performance deteriorates markedly at learning rates above 2×10^{-4}. This behavior is consistent with the trends observed in Section 4.2, regardless of the model architectures. Hence, in subsequent experiments, we will restrict our focus to learning rates in the range 2×10^{-5} to 2×10^{-4}.

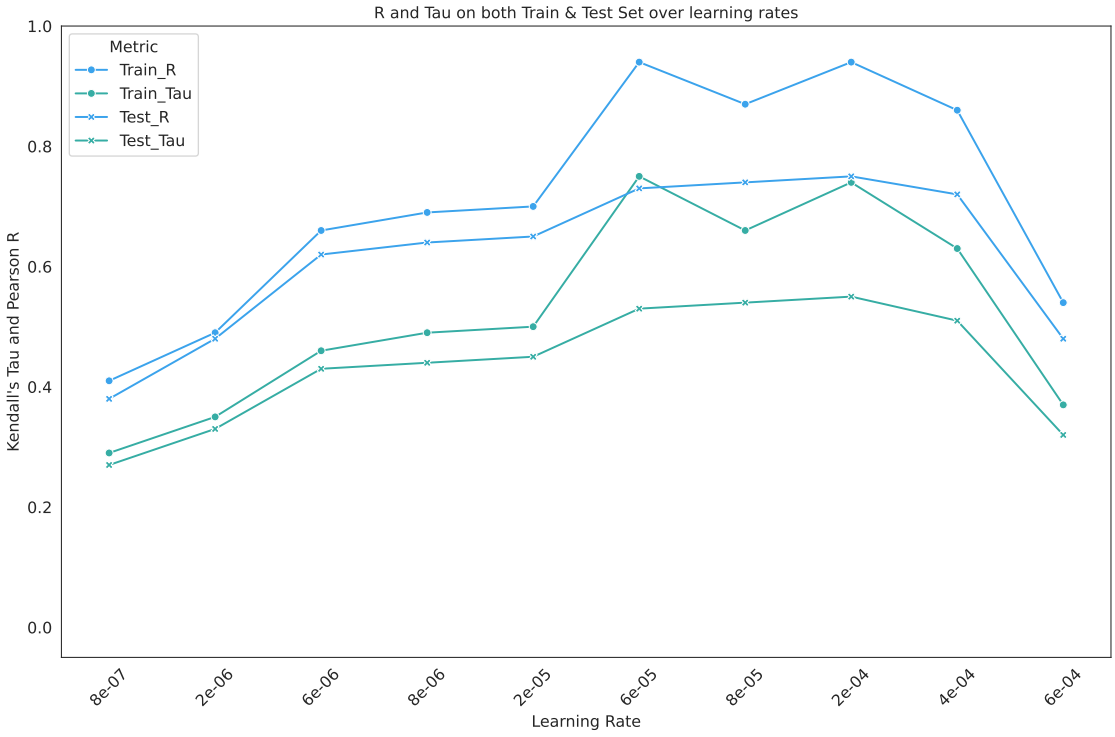

Figure 4.10: **Linear model** performance across various learning rates.

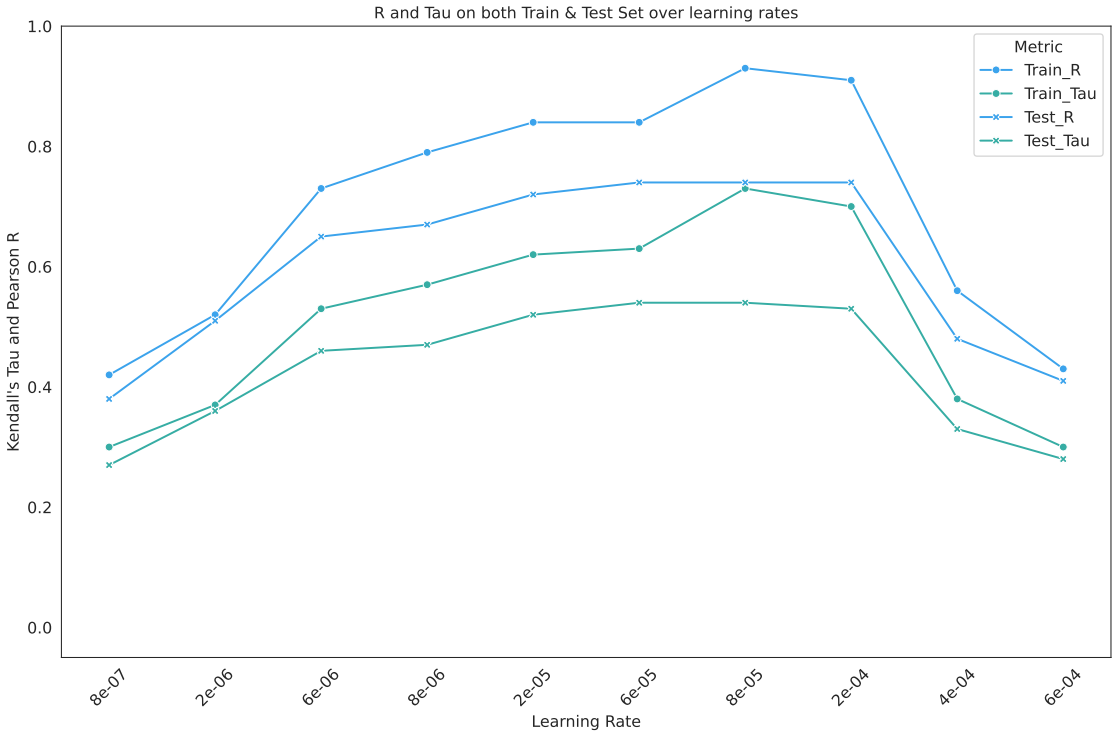

Figure 4.11: **STA model** performance across various learning rates.

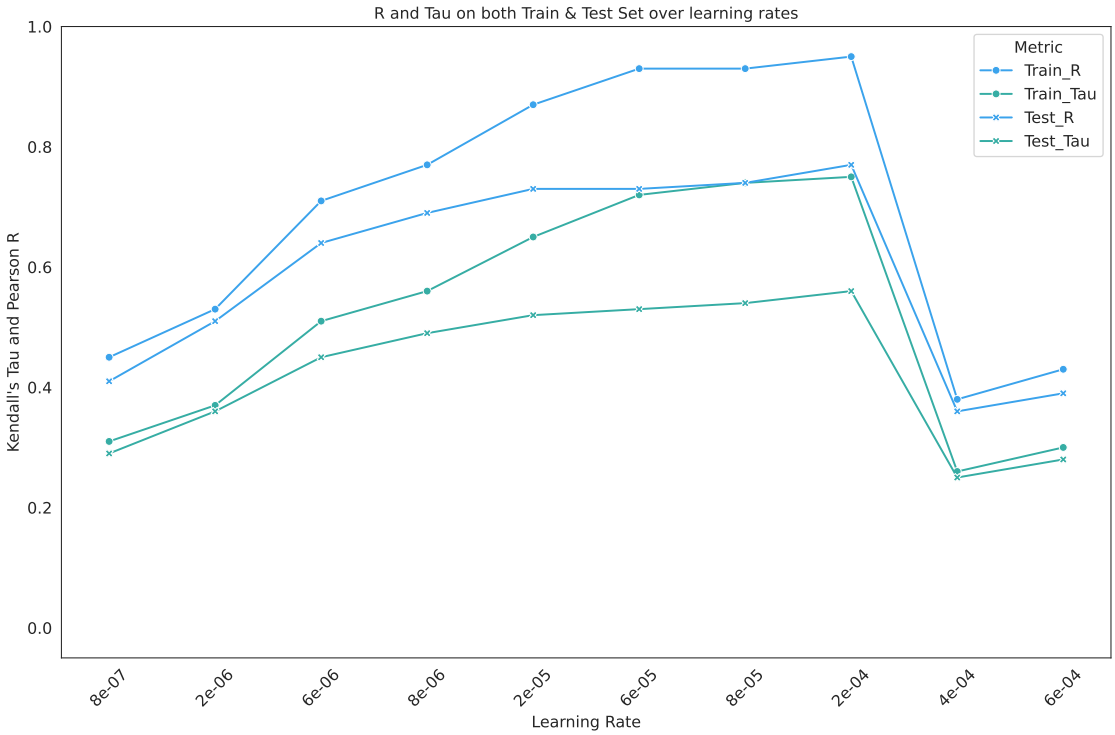

Figure 4.12: **GAT model** performance across various learning rates.

Training history across different architectures

Figure 4.13, 4.14, and 4.15 show the training history regarding the training loss and validation loss of three model architectures (Linear, STA, GAT) with the best learning rates. From the above figure, it is evident that owing to the incorporation of an early stopping mechanism, the epoch at which each model terminates is not uniform across architectures. Take GAT model as an example, compared to the training and validation loss trajectories when using smaller size of training data per epoch (Figure 4.6) versus using bigger size of training data per epoch (Figure 4.15) of the full dataset per epoch. On the Figure 4.6 when training the model with 10 percent of the full training data, the model's training loss starts at approximately 7.2 and decreases slowly over 300 epochs, while the validation loss follows a similar shallow decline but exhibits larger oscillations and plateaus around 1.4–1.6, indicating a pronounced gap between training and validation performance and slower convergence. In contrast, the Figure 4.15 shows an initial training loss near 6.4 that drops much more steeply—falling below 1.0 by roughly 200 epochs—and a validation curve that is markedly smoother, converging to around 1.1 with minimal fluctuations. Owing to the early stopping mechanism, the exact epoch at which training halts varies across different architectures.

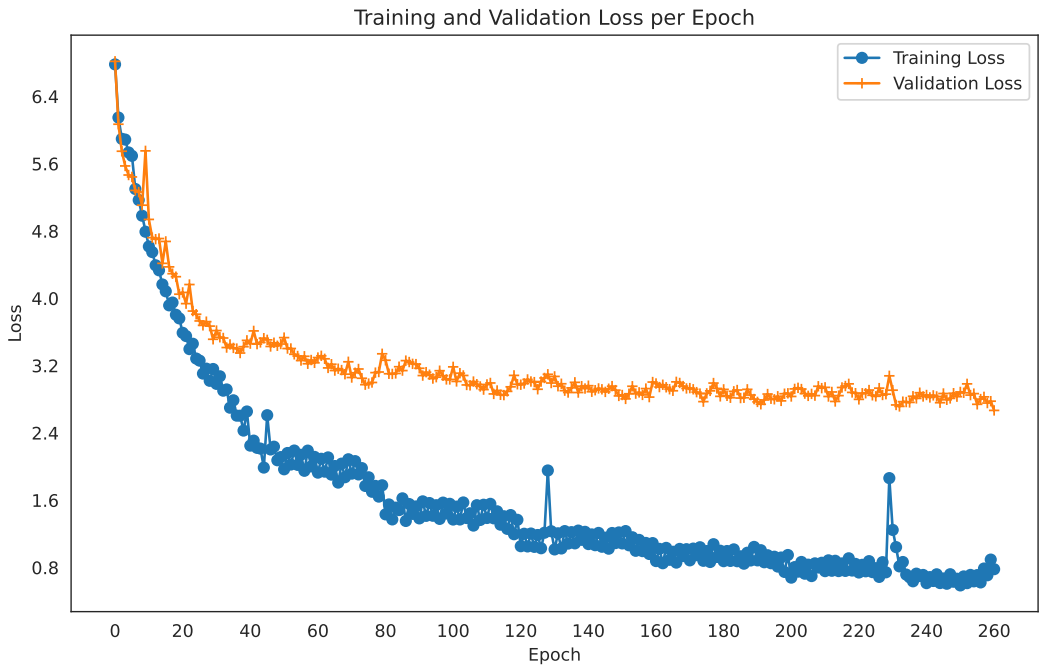

Figure 4.13: Training and validation loss per epoch for the **Linear model**.

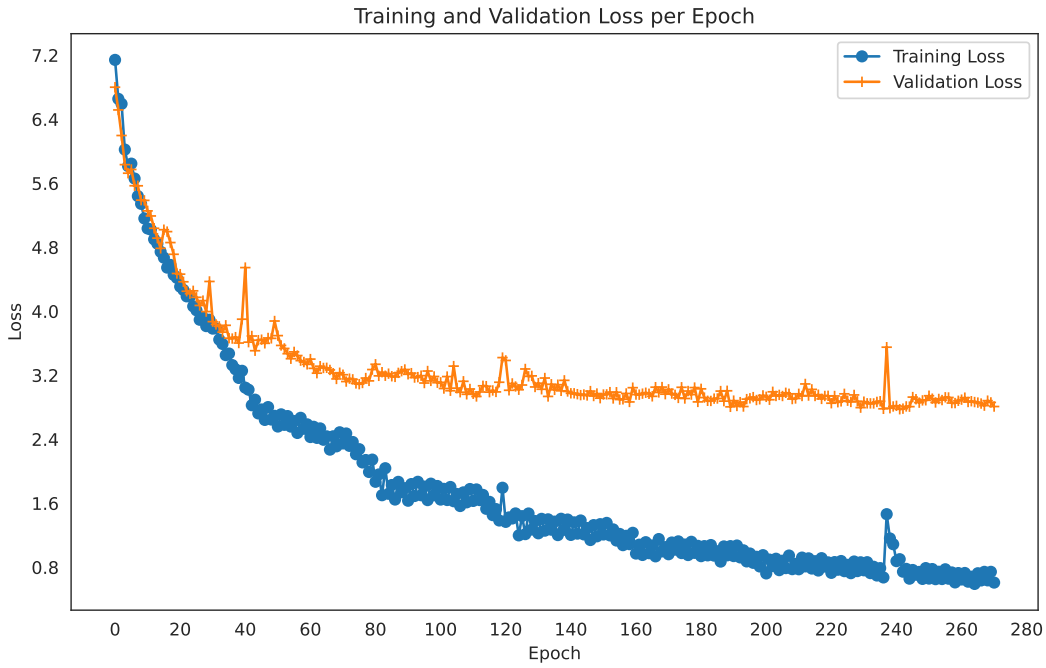

Figure 4.14: Training and validation loss per epoch for the **STA model**.

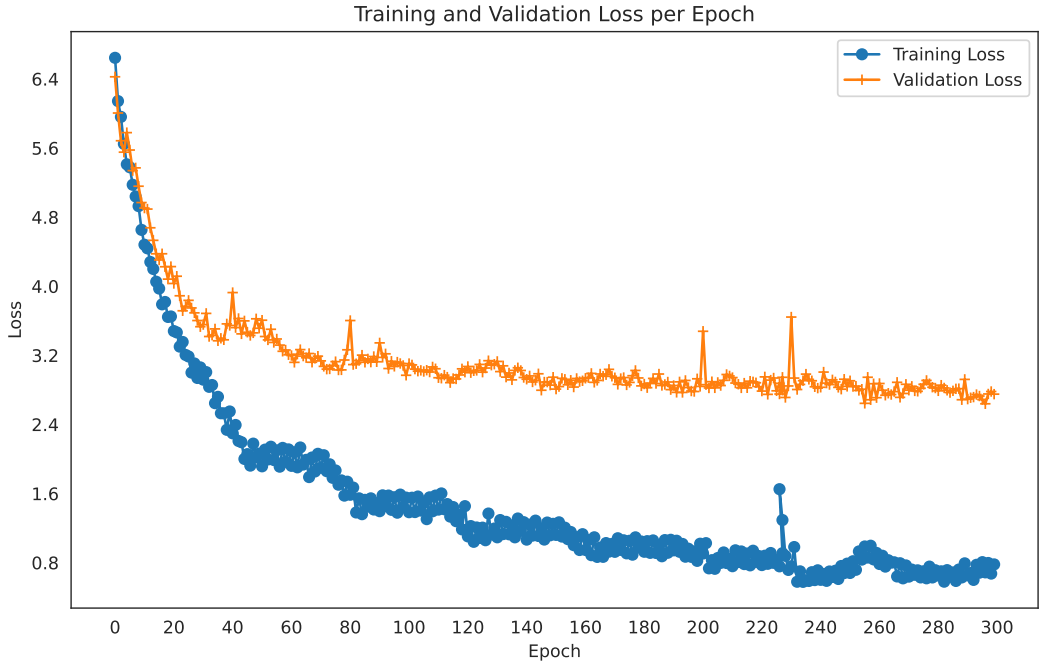

Figure 4.15: Training and validation loss per epoch for the **GAT model**.

Experimental vs. predicted scatter on training and test sets across archi-

tectures.
Figure 4.16, 4.17, and 4.18 show the performance on the test set regarding the Pearson's correlation coefficient (R) and Kendall's Tau (τ) across three model architectures (Linear, STA, GAT) with the best learning rates. From the figure above, compared with using samller size of the full-size training set per epoch, employing bigger sise of the training data at each epoch yields improved fit across all three models on both the training and test sets. However, the models exhibit pronounced overfitting, as evidenced by the divergence between the training and validation loss curves. Moreover, the proportions of False Positives (FP)—instances where the experimental values lie between 0 and 4 but are predicted to exceed 4—and False Negatives (FN)—instances where the experimental values exceed 4 but are predicted to be at most 4—do not appear to decrease significantly on either the training or test sets. Consequently, in subsequent experiments, we will investigate the use of a Learning Rate Scheduler (LRS) to determine whether it can enable the models to learn these edge cases more effectively.

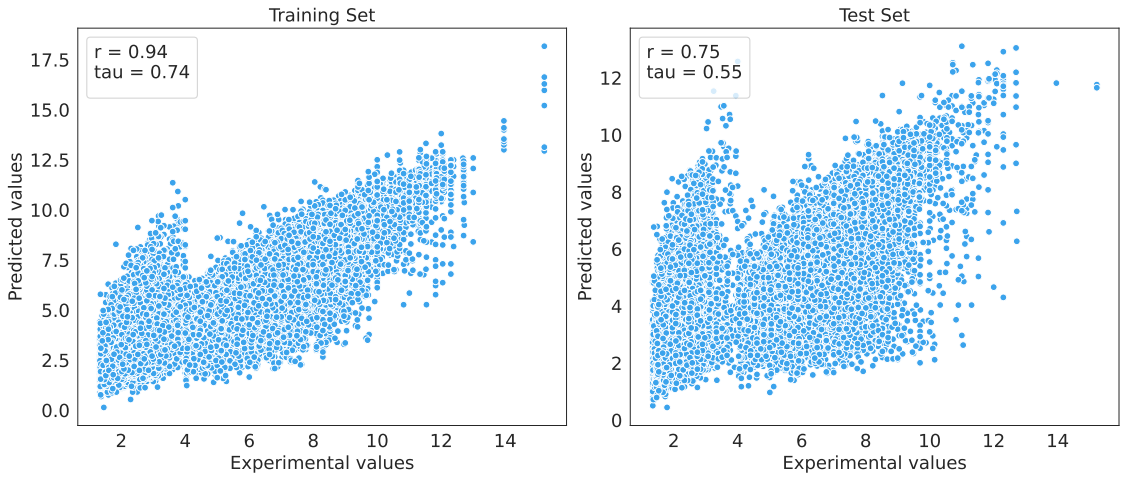

Figure 4.16: Experimental vs. predicted scatter on the train and test set for the **Linear model**.

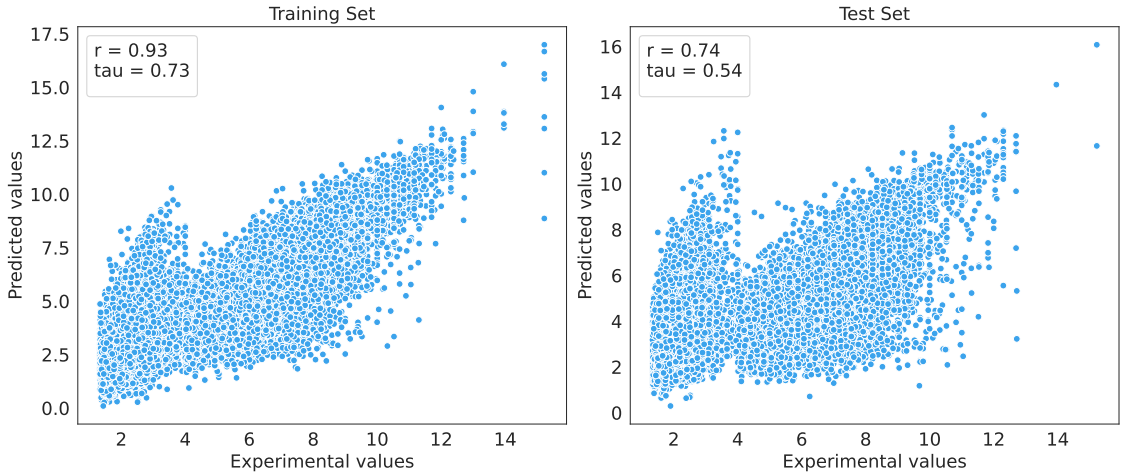

Figure 4.17: Experimental vs. predicted scatter on the train and test set for the **STA model**.

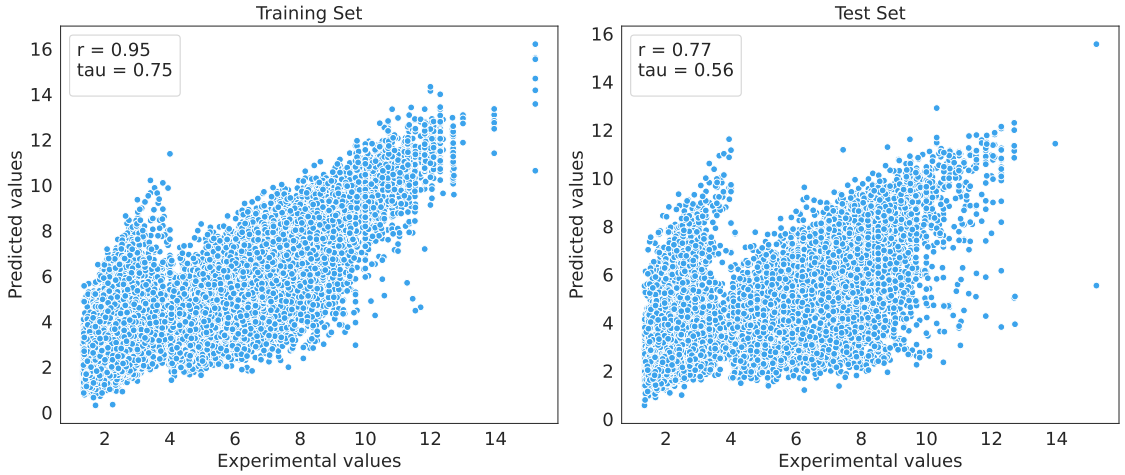

Figure 4.18: Experimental vs. predicted scatter on the train and test set for the **GAT model**.

4.2.2 Optimizing Training with Adaptive Learning Rate Schedulers

In this section, we investigate whether employing learning rate schedulers can enhance the performance of the Linear and GAT model architectures. We evaluate two schedulers from PyTorch: Cosine Annealing with Warm Restarts (CosineAnnealingWarmRestarts) and Cyclic Learning Rate (CyclicLR). The Cosine Annealing with Warm Restarts scheduler periodically resets the learning rate to reinvigorate the optimization process and avoid premature convergence. In contrast, the Cyclic Learning Rate scheduler oscillates the learning rate between predefined bounds, effectively mitigating local minima and promoting a more thorough exploration of the parameter space. For details on the operational mechanisms of both learning rate

schedulers, refer to Chapter 2.5. We aim to employ both learning rate schedulers to dynamically adjust the learning rate at each optimization step, thereby mitigating overfitting and accelerating convergence. All other experimental settings remain constant: the models are trained for 300 epochs.

Model performance across different architectures

Table 4.3: Model performances (Pearson's R and Kendall's τ) on train and test sets for various base learning rates with Cosine Annealing with Warm Restarts Learning Rate Scheduler.

Method	Base Learning Rate	Train		Test	
		R	τ	R	τ
Linear	2×10^{-4}	0.84	0.62	0.72	0.51
	8×10^{-5}	0.70	0.49	0.63	0.44
	6×10^{-5}	0.83	0.62	0.70	0.50
	2×10^{-5}	0.73	0.52	0.65	0.45
GAT	2×10^{-4}	0.95	0.75	0.73	0.54
	8×10^{-5}	0.95	0.75	0.75	0.55
	6×10^{-5}	0.95	0.75	**0.77**	**0.56**
	2×10^{-5}	0.94	0.74	0.76	0.55

Table 4.4: Model performances (Pearson's R and Kendall's τ) on train and test sets for various base learning rates with Cyclic Learning Rate Scheduler.

Method	Circle	Base Learning Rate	Train		Test	
			R	τ	R	τ
Linear	6 epochs	2×10^{-5}	0.92	0.72	**0.79**	**0.58**
	9 epochs	2×10^{-5}	0.90	0.68	0.78	0.57
GAT	6 epochs	2×10^{-5}	0.89	0.67	0.78	0.57
	9 epochs	2×10^{-5}	0.91	0.70	**0.79**	**0.58**

Tables 4.3 and 4.4 summarize the performance of the models, measured using Pearson's R and Kendall's τ, on both training and test. Table 4.3 presents results using the Cosine Annealing with Warm Restarts (CAWR) learning rate scheduler, while Table 4.4 reports performance under the Cyclic Learning Rate (CLR) scheduler.

Several key observations can be drawn from these results. First, the GAT model consistently outperforms the Linear model across both scheduling strategies and datasets. For instance, under the CAWR setup, the GAT model achieves its highest test performance at a base learning rate of 6×10^{-5}, with $R = 0.77$ and $\tau = 0.56$, significantly outperforming the Linear model at the same learning rate ($R = 0.70$,

$\tau = 0.50$). This clearly demonstrates the superior capacity of GAT in capturing complex structural information within the data.

Second, the choice of base learning rate has a substantial impact on model performance. In the case of CAWR, the optimal test performance for GAT is observed at a moderate base learning rate of 6×10^{-5}, indicating a critical balance between convergence speed and generalization. In contrast, the Linear model exhibits more variable behavior, with no single learning rate consistently yielding superior performance across both training and test phases.

Moreover, the results using the CLR scheduler (Table 4.4) reveal that both GAT and Linear models tend to exhibit slightly better test set performance compared to their CAWR counterparts. Notably, the GAT model reaches its overall peak performance at a circle of 9 and a base learning rate of 2×10^{-5}, achieving $R = 0.79$ and $\tau = 0.58$, the highest performance observed across all configurations.

In summary, these findings emphasize the critical role of learning rate and scheduling strategy selection, particularly for more complex architectures like GAT, where these hyperparameters can significantly influence the overall model performance.

Experimental vs. predicted scatter on training and test sets across architectures.
To further investigate the relationship between experimental and predicted values, scatter plots for the training and test sets were analyzed. Figures 4.19 and 4.20 illustrate the data distributions under the Cyclic Learning Rate (CLR) scheduler for the Linear model with a circle of 6 and the GAT model with a circle of 9, respectively.

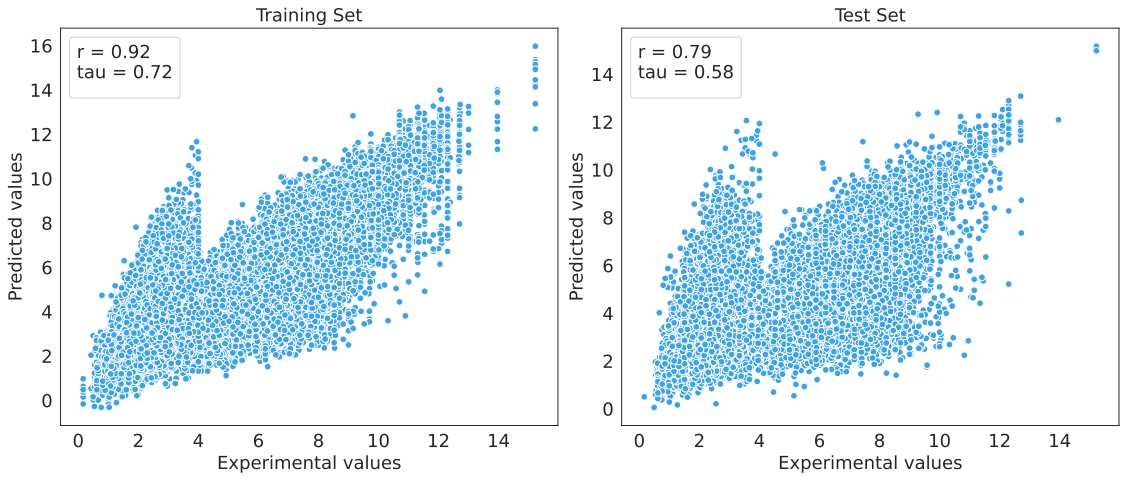

Figure 4.19: Experimental vs. predicted scatter on the train and test set for the **Linear model** trained with Cyclic Learning Rate Scheduler.

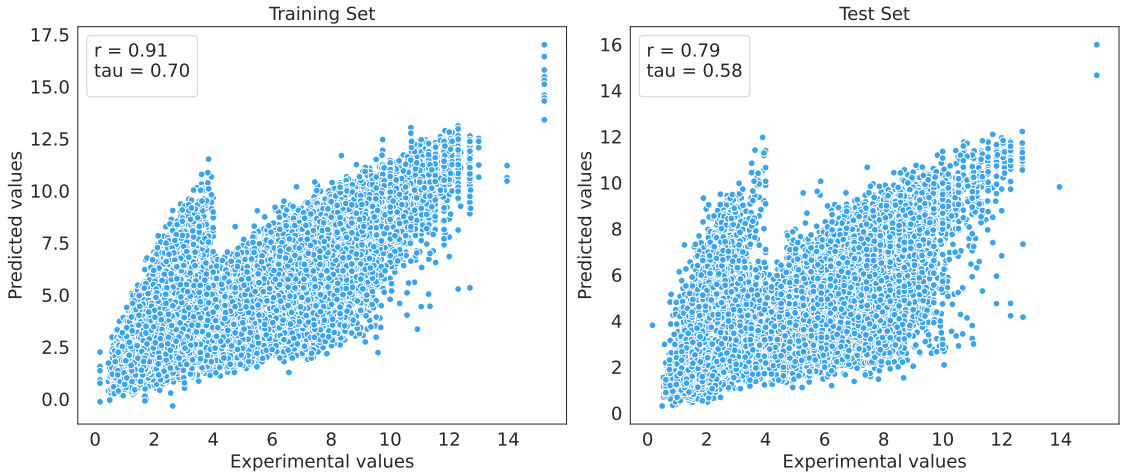

Figure 4.20: Experimental vs. predicted scatter on the train and test set for the **GAT model** trained with Cyclic Learning Rate Scheduler.

A closer examination of these plots reveals that both models exhibit improved overall fit on the training and test sets, suggesting that the chosen learning rate strategy effectively captures the underlying data patterns. However, despite this improved fit, the proportion of false positives (FP) and false negatives (FN) has not been significantly reduced, indicating that certain errors remain persistent across the different configurations.

4.3 Generalizability Analysis - Clustered Cross-Validation (CCV)

Clustered Cross-Validation (CCV) is a specialized cross-validation technique designed for datasets with inherent cluster structures. Unlike traditional k-fold cross-validation, which randomly splits individual samples into training and test sets, CCV groups data points into clusters and ensures that entire clusters, rather than individual samples, are allocated to different folds. This approach significantly reduces the risk of information leakage, where samples from the same cluster might otherwise appear in both training and test sets, leading to overly optimistic performance estimates.

The CCV process typically begins by identifying clusters based on domain knowledge or unsupervised clustering algorithms, such as k-means or hierarchical clustering. These clusters are then randomly partitioned into k folds, each serving as a test set in turn, while the remaining $k-1$ folds form the training set. By evaluating the model on these distinct clusters, CCV provides a more robust estimate of generalization performance, as it reduces the influence of cluster-specific correlations that might otherwise bias the evaluation.

In this section, we will analyze the results of CCV to assess the generalization capabilities of the models developed in this study. This approach aims to provide a more realistic assessment of model robustness by accounting for the complex

relationships within clusters, thereby offering a more accurate reflection of real-world performance.

Table 4.5: Clustered Cross-Validation (CCV) performance of Linear and GAT Models (Pearson's R and Kendall's τ) on train and test sets.

Linear Model

Cluster	Train		Test	
	R	τ	R	τ
C0	0.99	0.90	0.41	0.28
C1	0.97	0.86	0.38	0.24
C2	0.98	0.89	0.38	0.25
Average	0.98	0.88	0.39	0.26

GAT Model

Cluster	Train		Test	
	R	τ	R	τ
C0	0.99	0.93	0.42	0.28
C1	0.99	0.93	0.38	0.23
C2	0.99	0.93	0.38	0.26
Average	0.99	0.93	0.39	0.26

4.4 Results Analysis

In this section, we will conduct a detailed analysis of the results, focusing primarily on the model's performance through the examination of False Positives (FP) and False Negatives (FN). This approach allows us to gain deeper insights into the model's strengths and weaknesses, highlighting areas where it may overestimate or underestimate outcomes, thereby providing a more comprehensive understanding of its predictive accuracy.

4.4.1 Analysis of False Positives

In this section, we will conduct a detailed analysis of the False Positives (FP) identified in the model's predictions. As illustrated in Figures 4.19 and 4.20, these points form a sharp cluster located in the upper left corner. To gain deeper insights into the distribution of these points, we will closely examine the associated receptors and ligand names, with a particular focus on those with experimental values both less than 4 and greater than 4. Additionally, we will analyze the interaction fingerprints of these data points to investigate their potential impact on overall model performance.

Figure 4.21 presents the False Positives (FP) with experimental values less than 4 but predicted values greater than 4, as well as their corresponding receptors and ligands that share the same identities when their experimental values exceed 4. The six subplots in Figure 4.21 provide a comprehensive overview of the key factors influencing the distribution of False Positives (FP) within the model's predictions. In each subplot, we selectively remove certain data points and subsequently recalculate the Pearson correlation coefficient (R) and Kendall's Tau (τ) to investigate which subsets of data have the most significant impact on the overall model performance.

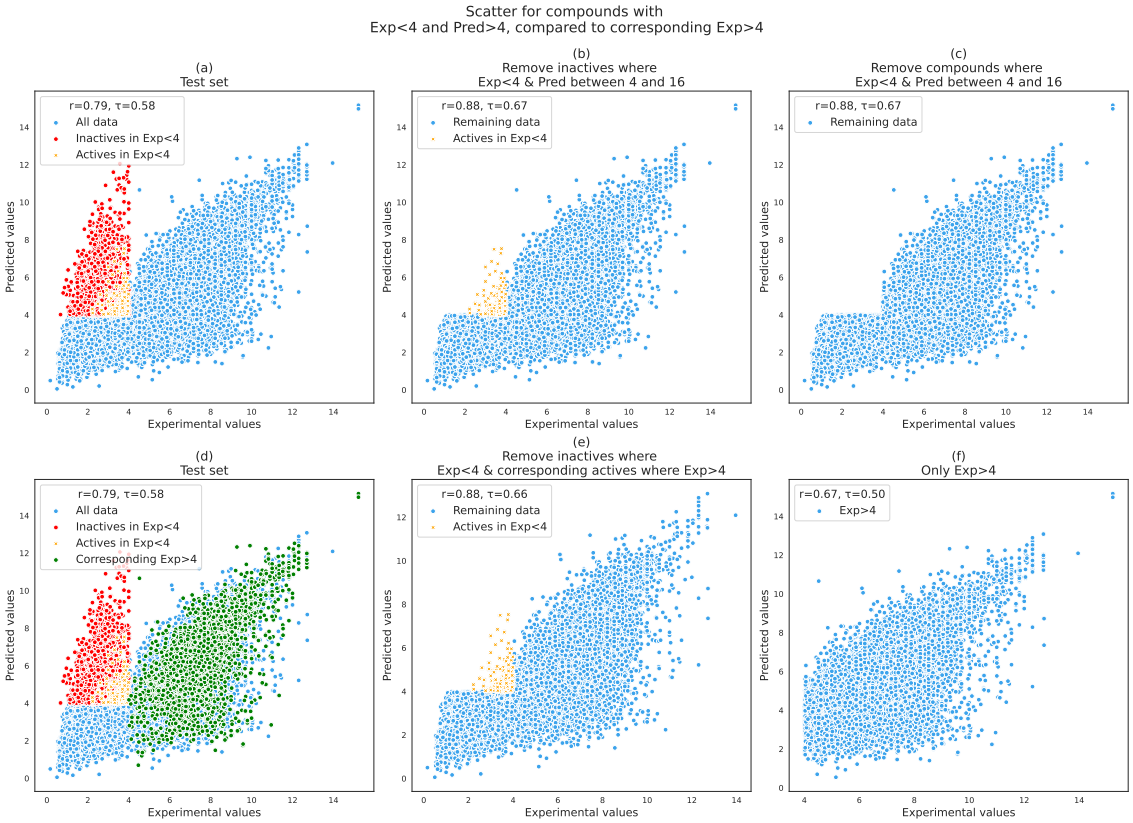

Figure 4.21: Scatter plot for compounds with Exp < 4 and Pred > 4, along with their corresponding compounds with Exp > 4.

The six subplots in Figure 4.21 provide a detailed examination of the distribution of False Positives (FP) in the model's predictions and their impact on overall performance. Each subplot illustrates a different perspective on how specific subsets of data influence the calculated Pearson correlation coefficient (R) and Kendall's Tau (τ):

- **Subplot (a):** This subplot presents the original test set, where the red and orange regions highlight the FP instances. Specifically, the red points represent data with experimental values less than 4 that were labeled as inactive in the CrossDocked2020 dataset, while the orange points correspond to data with experimental values less than 4 but labeled as active in the CrossDocked2020 dataset. Under these conditions, the model achieved an R of 0.79 and a τ of 0.58, respectively.
- **Subplot (b):** In this subplot, data points with experimental values less than 4 and labeled as inactive (the red points in subplot (a)) have been removed, while the orange points (active instances) are retained. Recalculating the model's performance on this modified set resulted in an R of 0.88 and a τ of 0.67, respectively. This represents a relative improvement of approximately 11.4% in R and 15.5% in τ compared to subplot (a).
- **Subplot (c):** This subplot shows all FP data points (both red and orange)

from subplot (a) are removed before recalculating the model's performance. Interestingly, the resulting R and τ are the same as in subplot (b) ($R = 0.88$, $\tau = 0.67$), indicating that the impact of the orange (active) points is minimal, likely due to their small sample size.

- **Subplot (d):** This subplot highlights the FP data points along with their corresponding receptor-ligand pairs that share the same identities but have experimental values greater than 4. These related points are shown in green, representing potentially meaningful connections between FPs and correctly predicted active cases.
- **Subplot (e):** In this subplot, the FP points (red) and their corresponding green points (those with matching receptor-ligand pairs but higher experimental values) are removed. Despite this additional filtering, the recalculated R and τ remain at 0.88 and 0.66, respectively, similar to subplot (b), suggesting that these green points have minimal impact on overall model performance, likely because they were already well predicted.
- **Subplot (f):** This subplot shows the model's performance after removing all data points with experimental values less than 4. The recalculated R and τ are 0.67 and 0.50, respectively, indicating a significant decline in performance. This finding highlights the critical importance of low experimental value data for accurate model evaluation.

Overall, Figure 4.21 reveals several critical insights into the impact of false positives (FP) on model performance. Removing inactive FP points, as seen in the transition from subplot (a) to subplot (b), leads to a substantial improvement in the model's correlation metrics, reflecting the strong influence of these errors on overall predictive accuracy. However, further removing the remaining FP points, as demonstrated in subplot (c), results in no additional gain, indicating that these active FP cases have a minimal impact, likely due to their smaller sample size. This trend continues in subplot (e), where removing the green receptor-ligand pairs associated with these FP cases also fails to significantly enhance performance, suggesting that these points were already well predicted by the model. In contrast, the sharp decline in correlation observed in subplot (f) after excluding all points with experimental values less than 4 underscores the critical importance of this data subset. These findings collectively highlight the disproportionate impact of low experimental value data on model performance, emphasizing the need for careful consideration of this subset in predictive modeling.

Additionally, we examined the distribution of these False Positives (FP) and their corresponding interactions with experimental values greater than 4 on a per-complex basis. Figure 4.22 presents the average number of interactions for these two groups, providing insights into the structural characteristics that may contribute to the observed prediction discrepancies. In this context, the red bars represent FP cases, which are defined as compounds with experimental values less than 4 but predicted values greater than 4, indicating potential overestimation by the model. In contrast, the green bars correspond to the same receptor-ligand pairs as these FP cases, but with experimental values greater than 4, reflecting instances where the same molecular

interactions are correctly predicted by the model as active. This distinction aligns with the scatter plot in Figure 4.21, where these two groups were similarly compared to assess the model's ability to differentiate between weak and strong binders based on their interaction profiles.

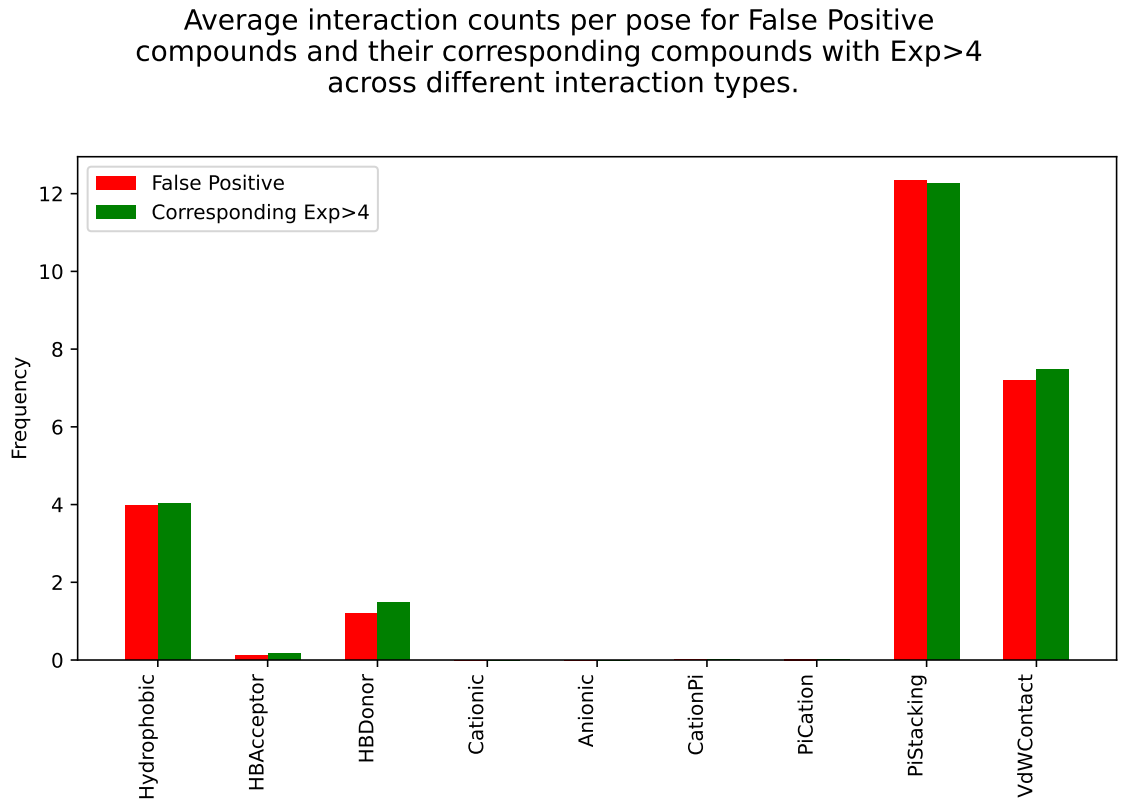

Figure 4.22: Average interaction counts per pose for False Positive compounds and their corresponding compounds with Exp > 4 across different interaction types.

The interaction distribution shown in Figure 4.22 reveals a high degree of similarity between the compounds with Exp < 4 and those with Exp > 4, suggesting that the model struggles to clearly differentiate these two groups based solely on their interaction profiles. Notably, both groups exhibit a dominant presence of π-stacking and hydrophobic interactions, which appear in nearly identical frequencies. This overlap indicates that these interaction types do not provide strong discriminative features for the model. In contrast, interactions like hydrogen bond donors (HBDonors) are slightly more frequent in the Exp > 4 group, potentially reflecting subtle but consistent structural differences. Other interaction types, such as anionic, cationic, and cation-π interactions, are largely absent in both groups, indicating a minimal role in distinguishing these compounds. The similarity in the distribution across these key interactions underscores the model's inherent challenge in accurately separating these data points, likely contributing to the observed prediction errors.

4.4.2 Analysis of False Negatives

In this section, we will perform a comprehensive examination of the False Negatives (FN) identified in the model's predictions. Unlike False Positives, which often result from overestimation, these points reflect cases where the model significantly underestimates the binding affinity, potentially missing important interactions. As illustrated in Figures 4.19 and 4.20, these FNs are typically distributed in regions where the experimental values are high, but the predicted values remain unexpectedly low. To gain a deeper understanding of these errors, we will closely inspect the associated receptors and ligand names, particularly those with experimental values both less than 4 and greater than 4. Furthermore, we will investigate the interaction fingerprints of these points to assess their contribution to the overall model performance, aiming to identify common features that may explain the model's underestimation in these cases.

Figure 4.23 illustrates the distribution of False Negatives (FN) in the model's predictions, specifically focusing on cases where the experimental values are greater than 4, but the predicted values fall below this threshold, indicating significant underestimation. Additionally, this figure includes the corresponding receptors and ligands that share the same identities as these FN cases when their experimental values are less than 4, providing a broader context for evaluating prediction consistency. The six subplots in Figure 4.23 present a systematic exploration of these FN cases, where data points are selectively removed to recalibrate the Pearson correlation coefficient (R) and Kendall's Tau (τ). This stepwise approach allows for a more nuanced assessment of how specific subsets of data influence overall model accuracy, shedding light on the molecular features and interaction patterns that may contribute to these underpredictions.

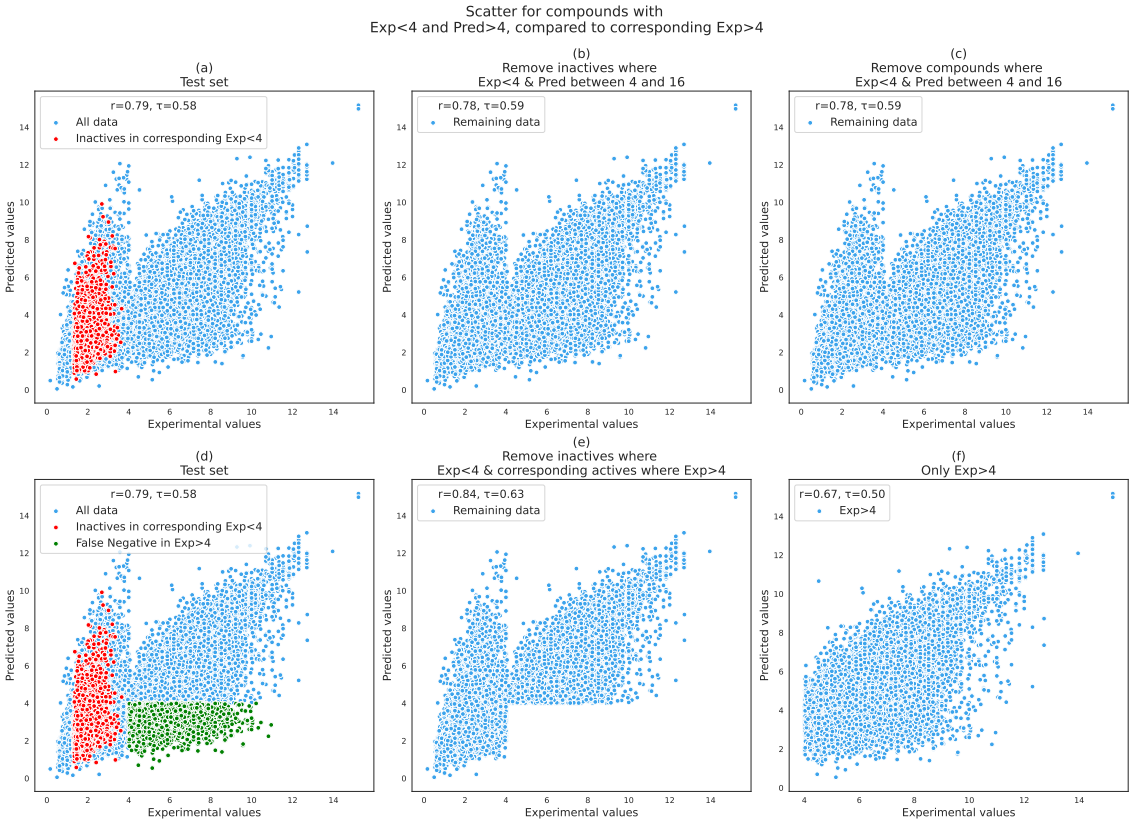

Figure 4.23: Scatter plot for compounds with Exp < 4 and Pred > 4, along with their corresponding compounds with Exp > 4.

The six subplots in Figure 4.23 provide a comprehensive examination of the distribution of False Negatives (FN) within the model's predictions and their impact on overall performance. Each subplot captures a different perspective on how various data subsets influence the calculated Pearson correlation coefficient (R) and Kendall's Tau (τ):

- **Subplot (a):** This subplot focuses on the corresponding points of the FN cases (represented in green in subplot (d)) that share the same receptor-ligand identities but have experimental values less than 4. These points are further categorized in red to indicate cases which are labeled as inactive in the CrossDocked2020 dataset. This highlights that, unlike the FP case, the corresponding points for FN in the experimental value range below 4 are predominantly inactive. The model originally achieves an R of 0.79 and a τ of 0.58, respectively.
- **Subplot (b):** Here, the red points representing inactive compounds with experimental values less than 4 have been removed, while the remaining data points are retained. Recalculating the model's performance after this filtering yields an R of 0.78 and a τ of 0.59, which is nearly identical to subplot (a), indicating that the removal of these inactive points has a minimal impact on overall model accuracy.

- **Subplot (c):** Given that there are no active data points with experimental values less than 4 in this subset, removing these points results in the same R and τ as in subplot (b), confirming that the inactive nature of these points dominates this region.
- **Subplot (d):** This subplot presents the original test set, where the green points represent the FN cases themselves (compounds with experimental values greater than 4 but predicted values less than 4). In addition, this subplot includes the corresponding points with the same receptor-ligand pairs as these FN cases but with experimental values less than 4, highlighted in red. Since these corresponding points are exclusively inactive in the CrossDocked2020 dataset, the overall model performance remains identical to subplot (a), with an R of 0.79 and a τ of 0.58, respectively.
- **Subplot (e):** In this subplot, both the green FN points and their corresponding red points (with experimental values less than 4 and matching receptor-ligand pairs) are removed, resulting in a recalculated R of 0.84 and a τ of 0.63, respectively. This reflects a significant performance improvement, with approximately a 6.3% increase in R and an 8.6% increase in τ compared to subplot (d). However, this gain is notably smaller than the corresponding improvement observed in the False Positive analysis (Figure 4.21 subplot (b)), which achieved approximately 11.4% in R and 15.5% in τ. This difference likely arises from the smaller number of inactive data points in the FN region, reducing the overall impact of this filtering step on model performance.
- **Subplot (f):** This subplot mirrors the approach taken in Figure 4.21 (f), where all data points with experimental values less than 4 are removed, resulting in a substantial decline in performance, with an R of 0.67 and a τ of 0.50, underscoring the critical importance of these low experimental value data for accurate model evaluation.

In contrast to the False Positive analysis, this examination of False Negatives (FN) reveals several distinct patterns. Subplots (a) through (c) demonstrate that removing inactive data points with low experimental values has a limited impact on overall model performance, resulting in only minor shifts in R and τ. This stability likely reflects the fact that these FN cases lack corresponding active points in the low experimental value region, reducing the influence of these filters. However, subplot (e) shows that removing both FN points and their associated low-experimental-value counterparts does provide a modest but noticeable improvement in correlation metrics, suggesting that even these inactive data can introduce subtle noise into the model. Nevertheless, this gain is smaller than the corresponding effect seen in the FP analysis, possibly due to the lower overall prevalence of such points in the FN subset. Finally, the significant drop in R and τ observed in subplot (f), after removing all data points with experimental values less than 4, highlights the critical importance of these low experimental value data for accurate model evaluation.

In addition, we analyzed the interaction profiles of these False Negatives (FN) and their corresponding receptor-ligand pairs with experimental values less than 4 on a per-complex basis. Figure 4.24 presents the average number of interactions for these

two groups, providing insights into the structural features that may contribute to the model's underestimation. In this context, the green bars represent FN cases, which are defined as compounds with experimental values greater than 4 but predicted values less than 4, indicating potential underestimation by the model. In contrast, the red bars correspond to the same receptor-ligand pairs as these FN cases, but with experimental values less than 4, reflecting instances where the same molecular interactions are consistently predicted as inactive. This comparison aligns with the scatter plot in Figure 4.23, where these two groups were similarly assessed to evaluate the model's ability to capture subtle structural differences that lead to FN errors.

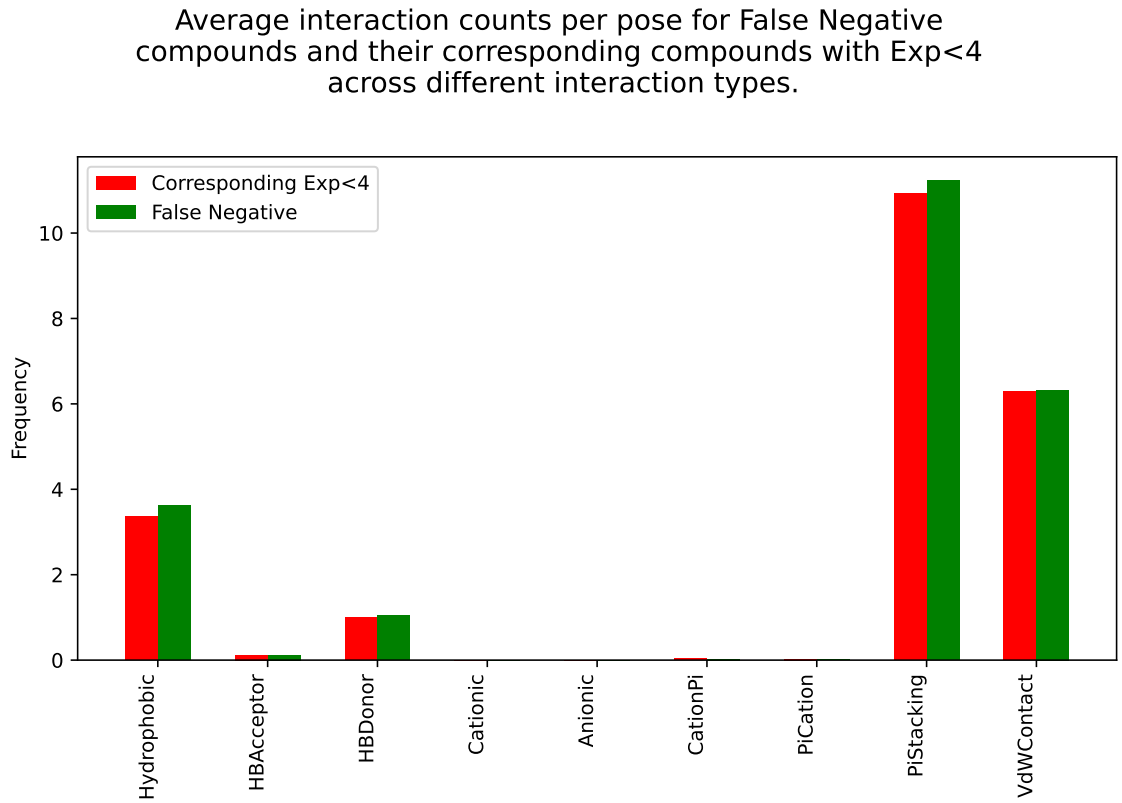

Figure 4.24: Average interaction counts per pose for False Negative compounds and their corresponding compounds with Exp < 4 across different interaction types.

The interaction profiles of False Negatives (FN) and False Positives (FP) reveal both shared patterns and distinct differences, reflecting the complex challenges the model faces in accurately capturing molecular interactions.

One major similarity is the dominance of π-stacking interactions in both FN and FP cases. This interaction type consistently appears as the most frequent across both error types, indicating that the model struggles to correctly estimate compounds characterized by strong stacking interactions. This is likely due to the complex, multi-dimensional nature of π-stacking, which involves both electrostatic and dispersion forces, making it particularly challenging to model accurately. Additionally, Van der Waals Contacts are also prominent in both FN and FP cases, reflecting their

foundational role in molecular recognition and binding affinity. The presence of these common interactions in both error types suggests that they are a critical factor in the model's overall predictive accuracy.

However, several notable differences emerge when comparing the FN and FP distributions. In the FN plot (Figure 4.24), π-stacking interactions are significantly more frequent among FN cases compared to their corresponding pairs with Exp < 4, indicating that the model tends to underestimate compounds where stacking interactions are a critical binding component. This imbalance suggests that the model is less sensitive to strong stacking interactions when making low-confidence predictions, potentially leading to systematic underestimation.

In contrast, the FP plot (Figure 4.22) shows a more balanced distribution of π-stacking interactions, where FP cases and their corresponding high-experimental-value pairs have nearly identical frequencies. This indicates that overestimation errors are less dependent on this interaction type, suggesting that the model's overconfidence in certain predictions is not as closely tied to stacking interactions as its underestimations are.

Hydrophobic interactions also exhibit distinct patterns. In the FN plot , these interactions are slightly more frequent among FN cases, suggesting that the model may not fully capture the relevance of hydrophobic contacts when predicting high-affinity complexes. In contrast, the FP distribution is nearly identical for this interaction type, implying that overestimation errors are less influenced by these non-polar contacts.

Van der Waals Contacts further highlight this distinction, with FN cases showing a noticeably higher frequency than their corresponding pairs, while the FP plot maintains a more balanced distribution. This suggests that the model's underestimation may be particularly affected by its handling of weaker, non-covalent interactions, which are often more context-dependent and less predictable than stronger, directional bonds.

Overall, these findings indicate that while both FN and FP cases are influenced by similar core interactions, the model's tendency to underestimate is more strongly associated with π-stacking and Van der Waals Contact interactions, whereas overestimation errors are more evenly distributed across different interaction types. This highlights the need for more refined interaction modeling, particularly in capturing the complex, multi-dimensional nature of π-stacking and subtle hydrophobic effects, to address these specific prediction challenges.

4.4.3 Analysis of Error Categories

In this section, we will evaluate the model's performance across different error categories, focusing on the distribution of over-predicted, well-predicted, and under-predicted data points. This analysis aims to identify the key factors contributing to each error type, including the relationship between predicted and experimental values, as well as the underlying interaction profiles that may influence model accuracy. By examining these distributions, we can gain insights into the specific interaction

features that lead to systematic overestimation or underestimation, providing a clearer understanding of the model's strengths and limitations.

The distribution of model predictions is illustrated in Figure 4.25, where data points are categorized into three groups based on a 1.5σ threshold, corresponding to an absolute error margin of approximately ± 2.47 units. In this plot, data points are classified as **Over-predicted** (displayed in red with 1,945 samples), **Well-predicted** (displayed in blue with 25,399 samples), and **Under-predicted** (displayed in green with 1,906 samples), reflecting different levels of agreement between predicted and experimental values. The dashed lines indicate the upper and lower threshold boundaries, defining the regions where predictions significantly deviate from the experimental measurements.

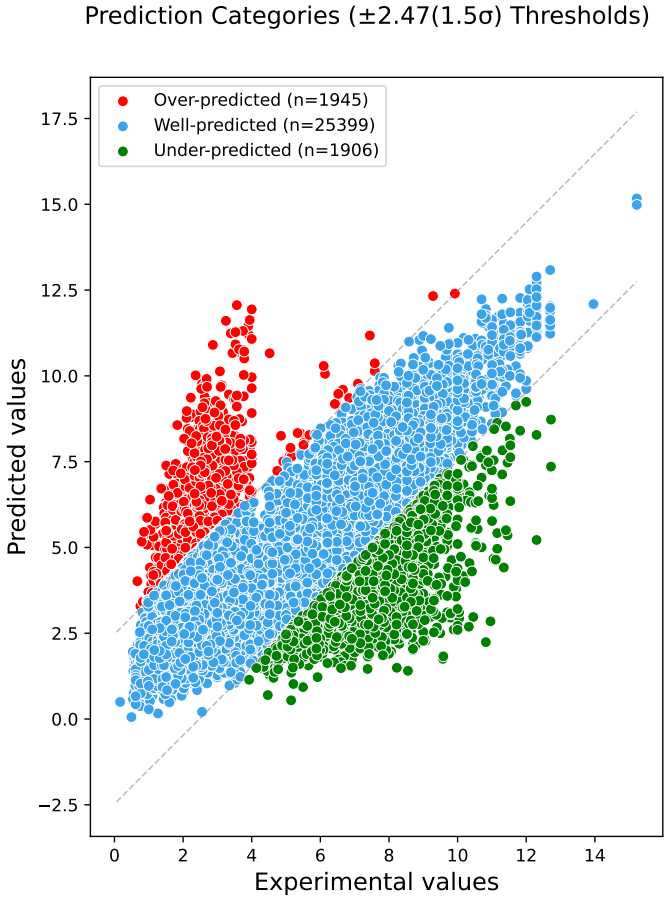

Figure 4.25: Scatter plot of predicted versus experimental values, categorized into Over-predicted (red), Well-predicted (blue), and Under-predicted (green) based on a 1.5σ threshold (± 2.47).

The interaction distribution across different prediction categories is presented in Figure 4.26. This plot illustrates the average interaction counts per pose for compounds classified as Over-predicted, Well-predicted, and Under-predicted based on the 1.5σ threshold. This plot reveals distinct interaction patterns across prediction categories.

π-stacking interactions are the most frequent across all categories, highlighting their critical role in binding affinity predictions. Van der Waals Contacts interactions also feature prominently but are particularly enriched in Under-predicted cases, suggesting that the model may systematically underestimate compounds where these weaker interactions are dominant. In contrast, Hydrophobic interactions appear more evenly distributed across all categories, indicating a less pronounced impact on prediction errors. Infrequent interaction types, such as Cationic, Anionic, CationPi, and PiCation, are consistently low across all groups, suggesting that these interactions have a minimal influence on the overall prediction performance.

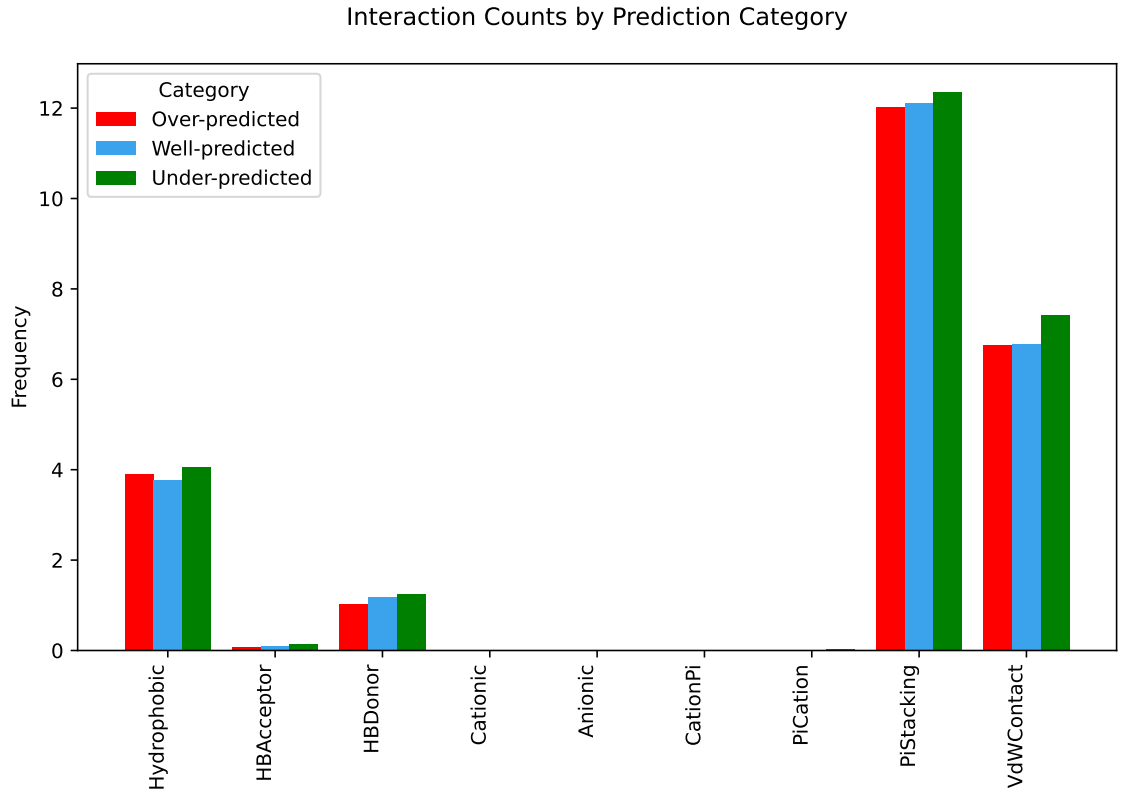

Figure 4.26: Average interaction counts per pose across different prediction categories: Over-predicted (red), Well-predicted (blue), and Under-predicted (green).

After examining the overall distribution of Over-predicted, Well-predicted, and Under-predicted compounds, we next focus on the well-predicted compounds, as they represent the majority of the data and provide valuable insights into the model's core predictive capabilities. To further explore the characteristics of these compounds, we divided the well-predicted region into three distinct subregions, as illustrated in Figure 4.27. In this figure, Region A (outlined in sky blue, includes 11,865 samples) includes compounds with experimental values greater than 4 that are well-predicted, representing the high-affinity binding cases accurately captured by the model. Region B (outlined in red, includes 1,359 samples) contains compounds with experimental values less than 4 that are over-predicted, indicating cases where the

model significantly overestimates the binding affinity. Finally, Region C (outlined in purple, includes 8,670 samples) represents compounds with experimental values less than 4 that are also well-predicted, reflecting accurate predictions for lower-affinity binders. The distribution of these regions is presented in Figure 4.27, which provides the exact compound counts for each region.

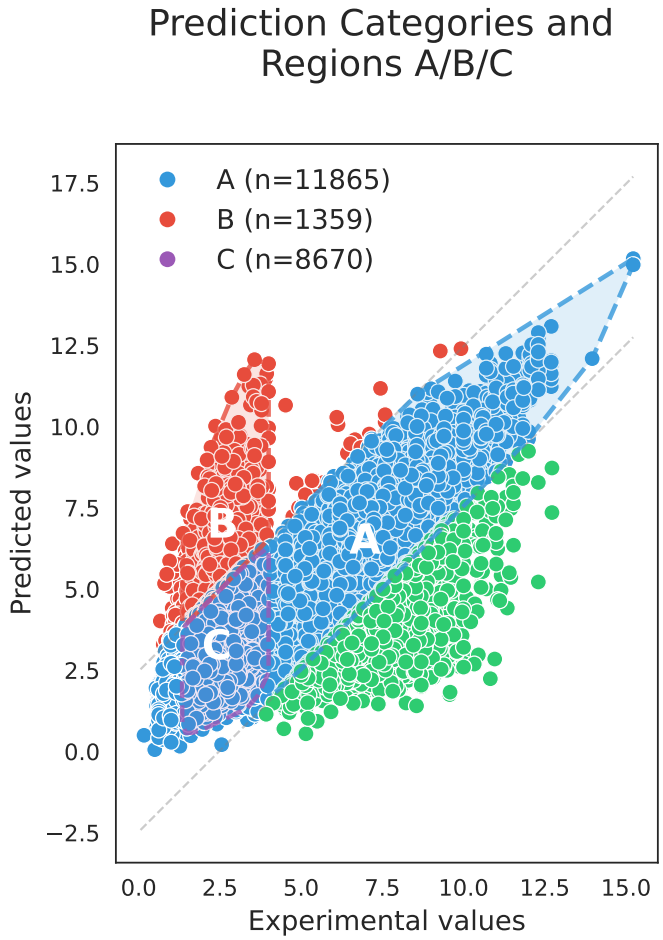

Figure 4.27: Scatter plot of predicted versus experimental values, categorized into Over-predicted (red), Well-predicted (blue), and Under-predicted (green) based on a 1.5σ threshold (± 2.47).

Figure 4.28 presents the distribution of average interaction counts for three defined regions (A, B, and C) within the well-predicted compounds. Region A (blue) corresponds to compounds with Exp > 4 that are well-predicted, Region B (red) includes over-predicted compounds with Exp < 4, and Region C (purple) represents well-predicted compounds with Exp < 4.

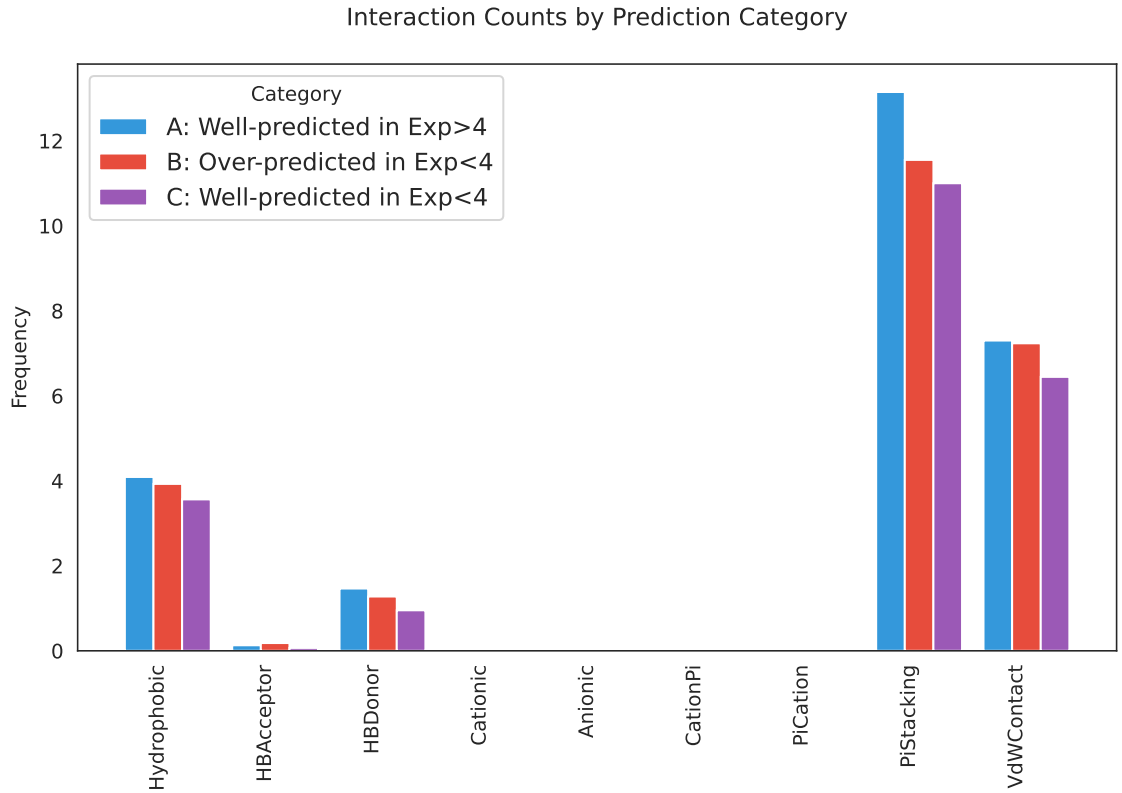

Figure 4.28: Average interaction counts per pose across three regions: Region A (blue) for compounds with Exp > 4 and accurate predictions, Region B (red) for over-predicted compounds with Exp < 4, and Region C (purple) for well-predicted compounds with Exp < 4.

When comparing this distribution to Figure 4.26, several critical differences emerge: First, π-stacking interactions are the most frequent across all regions, but their frequency is significantly higher in Region A, reflecting the importance of these interactions for high-affinity binders. In contrast, Regions B and C, both with Exp < 4, exhibit substantially lower π-stacking counts, indicating that low-affinity compounds are less dependent on this type of interaction. Second, Van der Waals Contact interactions are also more frequent in Region A, suggesting that high experimental value compounds benefit from a combination of strong π-stacking and weak non-covalent contacts. This pattern is consistent with the observation that Under-predicted compounds in Figure 4.26 also show a higher frequency of Van der Waals Contact interactions, potentially highlighting a shared challenge in accurately capturing these weak interactions. Finally, rare interaction types such as Cationic, Anionic, CationPi, and PiCation remain consistently low across all regions, suggesting minimal influence on the overall prediction performance, regardless of experimental value or prediction accuracy.

Overall, compounds in Region A rely more heavily on both π-stacking and Van der Waals Contact interactions, aligning with their higher experimental values, while compounds in Regions B and C exhibit weaker interaction profiles, potentially

contributing to the model's overestimation and underestimation errors.

To further investigate the combined effects of different prediction regions, we examine the distribution of interaction types in the merged regions A + C and B + C, as illustrated in Figure 4.29. This figure compares the individual regions (A, B, C) with their combinations (A + C and B + C), providing insights into how these merged groups influence the overall interaction landscape.

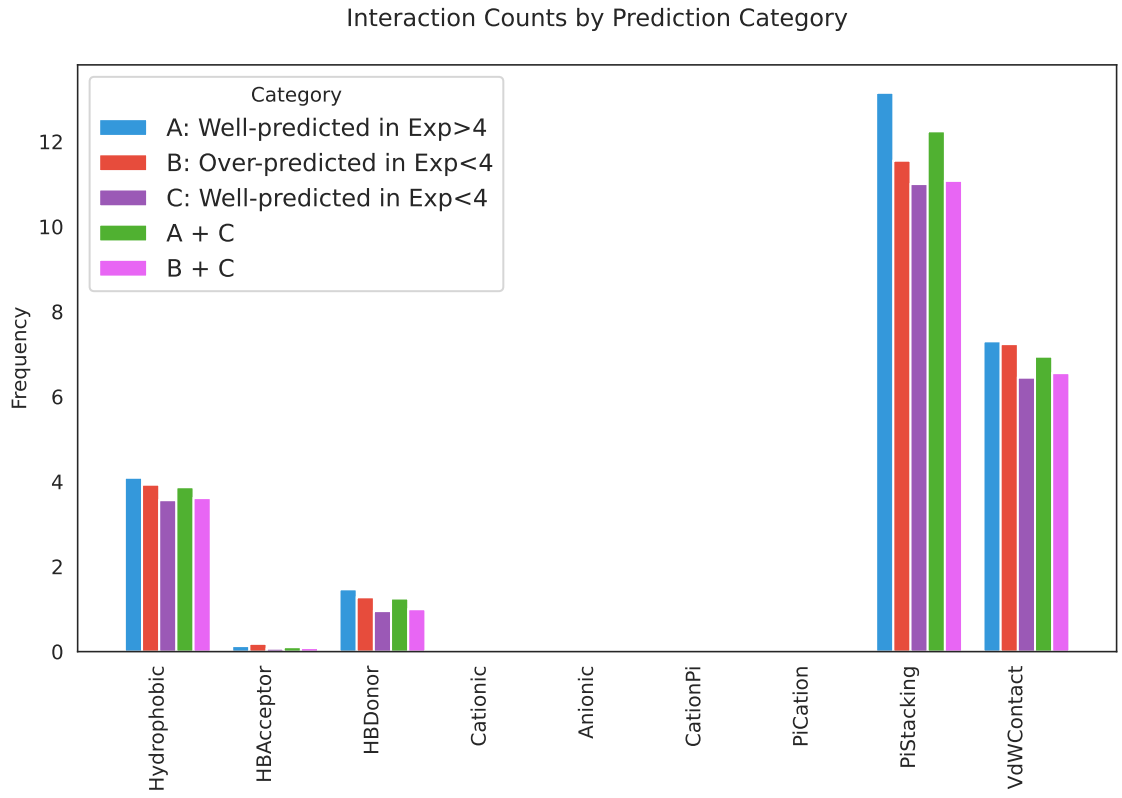

Figure 4.29: Average interaction counts per pose across the three defined regions and their combinations. The plot includes Region A (blue) for well-predicted compounds with Exp > 4, Region B (red) for over-predicted compounds with Exp < 4, and Region C (purple) for well-predicted compounds with Exp < 4. Additionally, the combined regions A + C (green) and B + C (pink) are included to highlight the interaction profiles when high and low experimental value compounds are considered together.

Examining this distribution reveals distinct patterns in the interaction profiles across different prediction regions. First, π-stacking interactions remain the most frequent across all groups, with Region A (Well-predicted, Exp > 4) exhibiting the highest frequency, followed closely by the combined A + C region. This consistency suggests that the inclusion of low experimental value compounds (Region C) does not significantly diminish the overall dominance of π-stacking in high-affinity binding cases. However, the B + C combination shows a noticeable decline in π-stacking, aligning more closely with the lower counts observed in Region B (Over-predicted,

Exp < 4), indicating a reduced reliance on this complex interaction type for low-affinity compounds. Second, Van der Waals Contact interactions follow a similar pattern, with Region A and the combined A + C region displaying the highest frequencies. This indicates that high experimental value compounds tend to rely more heavily on these weak, non-covalent contacts, even when mixed with lower experimental value compounds. In contrast, the B + C combination again exhibits a significant reduction, further highlighting the weaker interaction profiles of over-predicted, low-affinity compounds. Furthermore, Hydrophobic interactions, while less frequent overall, also show distinct patterns. Region A and A + C exhibit slightly higher counts, suggesting that high experimental value compounds maintain a stronger hydrophobic core even when mixed with lower experimental value cases. This contrasts with the B + C combination, where hydrophobic interactions are less prominent, potentially contributing to the model's tendency to overestimate binding affinities in this group. Finally, the rare interaction types (Cationic, Anionic, CationPi, PiCation) remain consistently low across all groups, reinforcing the earlier observation that these interactions have a minimal impact on overall prediction performance.

Overall, these findings suggest that the inclusion of high experimental value compounds (Region A) significantly shifts the interaction landscape, maintaining high π-stacking and Van der Waals Contact frequencies even in mixed groups. In contrast, the B + C combination consistently exhibits weaker interaction profiles, indicating a potential area for model refinement to better capture these low-affinity binding events. In addition, in Figure 4.29, we observed the interaction-type distributions for subsets A, B, C and their combinations. We further quantified these observations by noting that subset A (true positives with Exp > 4) contains 11,865 samples and subset C (true negatives with Exp < 4) contains 8,670 samples, for a combined total of 20,535 cases. On this combined set (A + C), the model achieves Pearson's $R = 0.93$ and Kendall's $\tau = 0.71$, respectively, demonstrating exceptionally high accuracy on these prototypical examples. By contrast, subset B (false positives with Exp $<$ 4 but Pred > 4) comprises only 1,359 samples; its interaction-type distribution is nearly identical to that of subset A. This similarity explains why the model struggles to distinguish these rare over-predictions from true positives. Importantly, however, the model still discriminates correctly predicted negatives (subset C) from the much larger set of positives with high reliability. These findings motivate our subsequent Top–N analysis, wherein we assess the impact of these error modes on the selection of high-confidence candidates.

4.4.4 Analysis of Top_N

In this section, we focus on evaluating the model's ability to identify the most confident predictions within the top-performing subsets of the data, referred to as "Top-N" analysis. This approach aims to assess the model's precision in identifying high-affinity binders, which is critical for practical applications like drug discovery, where only the most promising compounds are selected for further validation.

To achieve this, we define two key activity thresholds for classifying compounds as

"active" or "inactive" based on their experimental binding affinities (Exp) and model predictions (Pred). The first threshold is set at 4, aligning with common practice in binding affinity studies, while a more stringent threshold of 6 is also considered to capture only the strongest binders.

For each Top-N subset, we evaluate the following metrics:

- **Count**: The total number of active and inactive compounds within the selected top percentage.

- **TP (True Positives)**: Compounds that are correctly predicted as active (i.e., Pred > threshold) and are also experimentally confirmed as active (i.e., Exp > threshold). For example, with a threshold of 6, a compound with Pred = 7.5 and Exp = 6.8 is considered a true positive, reflecting accurate model identification of a strong binder.

- **FP (False Positives)**: Compounds that are predicted as active (i.e., Pred > threshold) but are experimentally confirmed as inactive (i.e., Exp $\leq$ threshold). For instance, if the threshold is set at 4, a compound with Pred = 5.2 but Exp = 3.8 would be classified as a false positive, as the model incorrectly predicts it as active despite its actual weak binding affinity.

- **FP Coverage (%)**: The proportion of false positives relative to all actual active compounds in the full dataset.

- **TP Coverage (%)**: The proportion of true positives within the subset, relative to all true positives in the full dataset.

This analysis is conducted across three key subsets: the top 1%, top 5%, and top 10% of the ranked predictions, allowing us to assess how well the model captures high-confidence active compounds at varying levels of selectivity.

For a more rigorous comparative evaluation, we provide two Top-N analysis tables based on different activity thresholds. Table 4.6 reports results where compounds with an experimental binding affinity (Exp) greater than 4 are considered active, representing a widely accepted cutoff in binding studies. In contrast, Table 4.7 presents results under a more stringent criterion, where only compounds with Exp > 6 are treated as active, emphasizing stronger binders. Each table summarizes the number of active and inactive compounds, as well as the False Positive (FP) and True Positive (TP) coverage rates, within the top 1%, 5%, and 10% subsets of ranked predictions. This comparative design enables a clearer assessment of the model's prioritization capability under varying stringency levels, highlighting its effectiveness in identifying high-affinity candidates while managing false discovery risks.

Table 4.6: Top-N Analysis (Threshold = 4)
Note:
FP (False Positives) are compounds where the predicted affinity is above the threshold (i.e., Pred $>$ 4) but the experimental affinity is below or equal to the threshold (i.e., Exp $\leq$ 4).
TP (True Positives) are compounds where both the predicted and experimental affinities are above the threshold (i.e., Pred $>$ 4 and Exp $>$ 4).

Top-N	**Count**	**TP**	**FP**	**FP-Coverage**	**TP-Coverage**
Top 1%	292	269	23	0.73%	2.27%
Top 5%	1462	1380	82	2.59%	11.62%
Top 10%	2925	2737	188	5.94%	23.05%
Total	2925	13796 (P)	15454 (N)	3165 (FP Total)	11872 (TP Total)

Table 4.7: Top-N Analysis (Threshold = 6)
Note:
FP (False Positives) are compounds where the predicted affinity is above the threshold (i.e., Pred $>$ 6) but the experimental affinity is below or equal to the threshold (i.e., Exp $\leq$ 6).
TP (True Positives) are compounds where both the predicted and experimental affinities are above the threshold (i.e., Pred $>$ 6 and Exp $>$ 6).

Top-N	**Count**	**TP**	**FP**	**FP-Coverage**	**TP-Coverage**
Top 1%	292	268	24	1.83%	3.61%
Top 5%	1462	1379	83	6.31%	18.56%
Top 10%	2925	2723	202	15.36%	36.66%
Total	2925	10523 (P)	18727 (N)	1315 (FP Total)	7428 (TP Total)

Comparing the two tables, several important trends emerge: First, the overall number of active compounds decreases significantly as the threshold increases from 4 to 6, reducing from 13,796 to 10,523, while the number of inactive compounds increases, reflecting a stricter definition of activity. Additionally, the TP-Coverage increases substantially at each Top-N level when using the higher threshold, rising from 2.27% to 3.61% at the Top 1% level and from 23.05% to 36.66% at the Top 10% level. This indicates that a higher threshold captures a greater proportion of true active compounds, consistent with its more selective definition. However, this gain in TP-Coverage comes at the cost of a significantly higher FP-Coverage, which more than doubles from 5.94% to 15.36% in the Top 10% subset, highlighting the increased risk of false positives as the threshold becomes more stringent.

Overall, these results suggest that while a higher activity threshold can improve the precision of Top-N selections, it also introduces a greater potential for misclassification, requiring careful consideration in practical applications.

5

Conclusion

5.1 Discussion

5.1.1 Learning-Rate Schedules as Regularizers

Beyond data volume, dynamic learning rates serve as implicit regularizers. Cosine Annealing with Warm Restarts (CAWR) periodically inject high-rate exploration, helping escape shallow minima, but can reintroduce instability if cycles are too frequent (Table 4.3). In contrast, the CyclicLR policy with a moderate amplitude (9) and conservative base LR (2×10^{-5}) consistently balances exploration and convergence, yielding our best generalization ($R = 0.79$, $\tau = 0.58$). This suggests that well-tuned oscillatory schedules offer a practical alternative to explicit regularization techniques such as weight decay or dropout, especially in data-scarce regimes.

5.1.2 Architecture-Dependent Sensitivities

Comparing Linear, STA, and GAT architectures under identical protocols reveals that model complexity alone does not guarantee better generalization. Under 10% sampling, Linear and GAT achieve identical test metrics (Table 4.1), indicating that the richer attention mechanisms in GAT require sufficient data to avoid under-utilization. Only when we combine 25% sampling with cyclic scheduling does GAT's structural inductive bias manifest, outperforming Linear by 4% in R. This underscores the necessity of aligning architectural complexity with training protocol: deep attention models demand both ample data exposure and stable optimization dynamics to realize their potential.

5.1.3 Insights from Error Subsets

Rare false positives—compounds with experimental affinity below 4 but predicted above 4—account for only 1,359 samples yet disproportionately degrade overall correlation. Interaction analysis (Figure 4.22) shows that these samples share nearly identical π-stacking and hydrophobic fingerprints with true positives. Removing these over-predicted cases recovers an 11% increase in Pearson R and a 15% increase in Kendall's τ, highlighting how distributional overlap can confound purely data-driven models and suggesting the need for auxiliary biophysical signals (e.g., ligand strain energy or protein conformational entropy) to disambiguate borderline cases.

Underestimated high-affinity complexes concentrate in regions dominated by strong π-stacking and extensive van der Waals contacts (Figure 4.24). The model's tendency to underpredict these interactions indicates that current message-passing layers may insufficiently capture multi-body dispersion and cooperativity effects. Incorporating specialized interaction modules or physics-informed priors could improve sensitivity to these high-energy binding modes and reduce systematic underestimation.

5.1.4 Practical Implications for Virtual Screening

Our Top–N analysis demonstrates that using a conservative affinity cutoff (Exp $>$ 4) yields a favorable precision–recall balance—5.9% false positives at 23.1% true-positive coverage—whereas stricter thresholds (Exp $>$ 6) substantially increase false positives (15.4%) despite higher recall. These results support a two-stage screening workflow: an initial broad pass at cutoff 4 to capture potential binders, followed by a secondary filter (e.g., docking rescoring or pharmacophore matching) to refine high-score candidates and minimize experimental follow-up on false positives.

5.1.5 Limitations and Outlook

The CrossDocked2020 dataset exhibits uneven representation across protein families and ligand sizes, which may bias model performance toward overrepresented targets. Future work should include validation on independent benchmarks (e.g., PDBbind) and deployment on chemotypes not seen during training to assess true transferability.

While chunked sampling addresses memory constraints, it increases per-epoch runtime by approximately 30%. Techniques such as stochastic weight averaging, progressive layering, or mixed-precision training could mitigate this overhead without compromising accuracy.

To address the identified interaction modeling gaps, we plan to integrate richer biophysical descriptors—such as polar-interaction fingerprints, ligand strain metrics, and dynamic solvent mappings—into our graph features. Additionally, attention-head interpretability analyses may guide targeted architecture refinements, improving both performance and explainability.

5.2 Conclusion

In this project, we introduced a graph neural network architecture for predicting protein–ligand binding affinities and evaluated it on the CrossDocked2020 dataset. We systematically explored how the choice of message-passing component—namely Linear, Set Transformer Aggregation (STA), or Graph Attention Network (GAT)—affects overall model performance. Additionally, we introduced a chunked-sampling training protocol that balances GPU memory constraints with distributional diversity, systematically compared Linear, STA, and GAT graph-based architectures, and evaluated two advanced learning-rate scheduling strategies. We found that training on 25% of the data per epoch with a CyclicLR scheduler yields the best generalization under the GAT model (Pearson's $R \approx 0.79$, Kendall's $\tau \approx 0.58$). Our fine-grained error

analysis revealed that the model achieves near-perfect correlation on typical examples (subset A+C, where *subset A* consists of true positives (experimental affinity4 and correctly predicted) and *subset C* consists of true negatives (experimental affinity<4 and correctly predicted), Pearson's $R = 0.93$ and Kendall's $\tau = 0.71$, respectively), while a small number of over-predicted (subset B) and under-predicted cases highlight areas for further refinement. Future work will focus on augmenting sparsely sampled affinity regions, integrating richer biophysical descriptors, and implementing targeted loss-function strategies to mitigate residual mispredictions. By embedding these enhancements into a real-world virtual screening pipeline with experimental validation, we aim to further improve hit rates and streamline lead discovery in structure-based drug design.

Bibliography

[1] R. Milo, "What is the total number of protein molecules per cell volume? a call to rethink some published values," *Bioessays*, vol. 35, no. 12, pp. 1050–1055, 2013.

[2] B. Besson, H. Eun, S. Kim, M. P. Windisch, H. Bourhy, and R. Grailhe, "Optimization of bret saturation assays for robust and sensitive cytosolic protein–protein interaction studies," *Scientific Reports*, vol. 12, no. 1, p. 9987, 2022.

[3] E. Kobayashi, A. Jin, H. Hamana, *et al.*, "Rapid cloning of antigen-specific t-cell receptors by leveraging the cis activation of t cells," *Nature Biomedical Engineering*, vol. 6, no. 7, pp. 806–818, 2022.

[4] C. Wu, Y. Xu, Q. He, *et al.*, "Ligand-induced activation and g protein coupling of prostaglandin f2α receptor," *Nature Communications*, vol. 14, no. 1, p. 2668, 2023.

[5] E. Armingol, A. Officer, O. Harismendy, and N. E. Lewis, "Deciphering cell–cell interactions and communication from gene expression," *Nature Reviews Genetics*, vol. 22, no. 2, pp. 71–88, 2021.

[6] Y. Mizukoshi, K. Takeuchi, Y. Tokunaga, *et al.*, "Targeting the cryptic sites: Nmr-based strategy to improve protein druggability by controlling the conformational equilibrium," *Science advances*, vol. 6, no. 40, eabd0480, 2020.

[7] P. C. Souza, S. Thallmair, P. Conflitti, *et al.*, "Protein–ligand binding with the coarse-grained martini model," *Nature communications*, vol. 11, no. 1, p. 3714, 2020.

[8] G. Rastelli, A. D. Rio, G. Degliesposti, and M. Sgobba, "Fast and accurate predictions of binding free energies using mm-pbsa and mm-gbsa," *Journal of computational chemistry*, vol. 31, no. 4, pp. 797–810, 2010.

[9] G. M. Morris, R. Huey, W. Lindstrom, *et al.*, "Autodock4 and autodocktools4: Automated docking with selective receptor flexibility," *Journal of computational chemistry*, vol. 30, no. 16, pp. 2785–2791, 2009.

[10] O. Trott and A. J. Olson, "Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of computational chemistry*, vol. 31, no. 2, pp. 455–461, 2010.

[11] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of molecular biology*, vol. 267, no. 3, pp. 727–748, 1997.

[12] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, "A fast flexible docking method using an incremental construction algorithm," *Journal of molecular biology*, vol. 261, no. 3, pp. 470–489, 1996.

[13] S.-Y. Huang and X. Zou, "Inclusion of solvation and entropy in the knowledge-based scoring function for protein- ligand interactions," *Journal of chemical information and modeling*, vol. 50, no. 2, pp. 262–273, 2010.

[14] A. V. Ishchenko and E. I. Shakhnovich, "Small molecule growth 2001 (smog2001): An improved knowledge-based scoring function for protein- ligand interactions," *Journal of medicinal chemistry*, vol. 45, no. 13, pp. 2770–2780, 2002.

[15] C. Zhang, S. Liu, Q. Zhu, and Y. Zhou, "A knowledge-based energy function for protein- ligand, protein- protein, and protein- dna complexes," *Journal of medicinal chemistry*, vol. 48, no. 7, pp. 2325–2335, 2005.

[16] P. J. Ballester and J. B. Mitchell, "A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking," *Bioinformatics*, vol. 26, no. 9, pp. 1169–1175, 2010.

[17] G.-B. Li, L.-L. Yang, W.-J. Wang, L.-L. Li, and S.-Y. Yang, "Id-score: A new empirical scoring function based on a comprehensive set of descriptors related to protein–ligand interactions," *Journal of chemical information and modeling*, vol. 53, no. 3, pp. 592–600, 2013.

[18] J. Gabel, J. Desaphy, and D. Rognan, "Beware of machine learning-based scoring functions on the danger of developing black boxes," *Journal of chemical information and modeling*, vol. 54, no. 10, pp. 2807–2815, 2014.

[19] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.

[20] H. Liu, Y. Jian, J. Hou, C. Zeng, and Y. Zhao, "Rnet: A network strategy to predict rna binding preferences," *Briefings in Bioinformatics*, vol. 25, no. 1, bbad482, 2024.

[21] J. Jumper, R. Evans, A. Pritzel, *et al.*, "Highly accurate protein structure prediction with alphafold," *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[22] A. W. Senior, R. Evans, J. Jumper, *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[23] H. Öztürk, A. Özgür, and E. Ozkirimli, "Deepdta: Deep drug–target binding affinity prediction," *Bioinformatics*, vol. 34, no. 17, pp. i821–i829, 2018.

[24] I. Wallach, M. Dzamba, and A. Heifets, "Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery," *arXiv preprint arXiv:1510.02855*, 2015.

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

[26] M. Sun, S. Zhao, C. Gilvary, O. Elemento, J. Zhou, and F. Wang, "Graph convolutional networks for computational drug development and discovery," *Briefings in bioinformatics*, vol. 21, no. 3, pp. 919–935, 2020.

[27] J. Gomes, B. Ramsundar, E. N. Feinberg, and V. S. Pande, "Atomic convolutional networks for predicting protein-ligand binding affinity," *arXiv preprint arXiv:1703.10603*, 2017.

[28] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[29] E. N. Feinberg, D. Sur, Z. Wu, *et al.*, "Potentialnet for molecular property prediction," *ACS central science*, vol. 4, no. 11, pp. 1520–1530, 2018.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, pp. 11–15, 2008. DOI: 10.1109/TNN.2008.2005605.

[31] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Liò, "Graph neural networks with adaptive readouts," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 746–19 758, 2022.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations (ICLR)*, pp. 1–12, 2018. DOI: 10.48550/arXiv.1710.10903. [Online]. Available: https://arxiv.org/abs/1710.10903.

[33] J. Fürnkranz, "Round robin rule learning," in *ICML*, vol. 1, 2001, pp. 146–153.

[34] J. Fürnkranz, "Round robin classification," *Journal of Machine Learning Research*, vol. 2, no. Mar, pp. 721–747, 2002.

[35] C. S. Doire and O. Okubadejo, "Interleaved multitask learning for audio source separation with independent databases," *arXiv preprint arXiv:1908.05182*, 2019.

[36] D. Mayo, T. R. Scott, M. Ren, *et al.*, "Multitask learning via interleaving: A neural network investigation," in *Proceedings of the annual meeting of the cognitive science society*, vol. 45, 2023.

[37] L. N. Smith, "Cyclical learning rates for training neural networks," *arXiv preprint arXiv:1506.01186*, 2015.

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

[39] E. B. Lenselink, N. Ten Dijke, B. Bongers, *et al.*, "Beyond the hype: Deep neural networks outperform established methods using a chembl bioactivity benchmark set," *Journal of cheminformatics*, vol. 9, pp. 1–14, 2017.

[40] I. Kufareva, A. V. Ilatovskiy, and R. Abagyan, "Pocketome: An encyclopedia of small-molecule binding sites in 4d," *Nucleic acids research*, vol. 40, no. D1, pp. D535–D540, 2012.

[41] A. Bakan, A. Dutta, W. Mao, *et al.*, "Evol and prody for bridging protein sequence evolution and structural dynamics," *Bioinformatics*, vol. 30, no. 18, pp. 2681–2683, 2014.

[42] D. R. Koes, M. P. Baumgartner, and C. J. Camacho, "Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise," *Journal of chemical information and modeling*, vol. 53, no. 8, pp. 1893–1904, 2013.

[43] J. Konc and D. Janežič, "Probis algorithm for detection of structurally similar protein binding sites by local structural alignment," *Bioinformatics*, vol. 26, no. 9, pp. 1160–1168, 2010.

A

Appendix 1