A Graph Convolutional Network Approach for $logK_d$ and $logK_i$ Prediction

Vivek Tara

May 2025

1 Abstract

2 Protein-Ligand Interactions

2.1 Physicochemical Mechanisms of Protein-Ligand Interaction

2.1.1 Protein-Ligand Binding Kinetics

Protein-ligand binding kinetics describe the process underlying the association between the protein and ligand, particularly by focusing on the rate at which these two partners bind to each other. In a simple instance, when a protein molecule P and a ligand molecule L with mutual affinity are mixed in a solution, the time-dependent association between them can be formulated as:

$$P + L \rightleftarrows PL \tag{1}$$

Where PL represents the protein-ligand complex, k_{on} and k_{off} are the kinetic rate constants that account for the forward binding and reverse dissociation reaction, respectively. The units of k_{on} and k_{off} are $M^{-1} \cdot s^{-1}$ and s^{-1} respectively.

At equilibrium, the forward binding reaction $P+L \rightarrow PL$ should be balanced by the reverse unbinding reaction $PL \rightarrow P + L$, and this is written:

$$k_{on}[P][L] = k_{off}[PL] \tag{2}$$

Here, the square brackets represent the equilibrium concentration of any molecular species. The binding constant K_b (in units of M^{-1}) is then defined by:

$$K_b = \frac{k_{on}}{k_{off}} = \frac{[PL]}{PL} = \frac{1}{K_d} \tag{3}$$

Where K_d (in units of M) is called the dissociation constant. Therefore, the fast binding rate accompanied by a slow dissociation rate will give a high/low binding/dissociation constant and hence, a high binding affinity.

Basic Concepts and Thermodynamic Relationships

A protein-ligand-solvent system is a thermodynamic system composed of the solute (i.e, the protein and ligand molecules) and the solvent (i.e the water and buffer ions). There are complex interactions and heat exchange among these substances; and the relationship between these substances and how heat transfer is related to various energy changes are dictated by the laws of thermodynamics.

Gibbs free energy, which is a thermodynamic potential (i.e a scalar quantity used to represent the thermodynamic state of a system) that measures the capacity of a thermodynamic system to do maximum or reversible work at a constant temperate and pressure (isothermal, isobaric), is one of the most important thermodynamic quantities for the characterization of the driving forces that dictate the associations between protein and ligands. In analogy with any spontaneous process, protein-ligand binding occurs only when the change in the Gibbs free energy ΔG of the system is negative when the system reaches an equilibrium state with constant pressure and temperature.

Because the protein-ligand association extent is determined by the magnitude of the negative ΔG, it can be considered that ΔG determines the stability of any given protein-ligand complex, or, alternatively the binding affinity of a ligand to a given acceptor. G is a function of the states of a system, thus ΔG is defined solely by the initial and final thermodynamic states, regardless of the pathway connecting these two states.

The standard binding free energy ΔG, which refers to the free energy change measured under the conditions of $1atm$ pressure, a temperature of $298K$, and the effective reactant (protein and ligand) concentrations of $1M$, is related to the binding energy constant K_b by the Gibbs relationship:

$$\Delta G = -RTlnK_b \tag{4}$$

Where R is the universal gas constant and T is the temperature in units of K. It is clear from equation 4 that a larger binding constant K_b is associated with a more negative standard free energy of binding, indicating the kinetic parameters, (the ratio of k_{on} and k_{off}) determines the thermodynamic properties of the complex, i.e, the stability of the complex and the binding affinity between the protein and the ligand.

The binding free energy (ΔG) at any moment in time during an association (not necessarily at standard-state condition) is given by:

$$\Delta G = \Delta G + RTlnQ \tag{5}$$

Where Q is the reaction quotient, defined as the ratio of the concentration of the protein-ligand complex to the product of the concentrations of the free protein and free ligand at any moment in time. When $Q = K_b$, an association reaction is at equilibrium, and $\Delta G = 0$. ΔG can also be parsed into its enthalpic and entropic contributions with the following fundamental equation:

$$\Delta G = \Delta H - T\Delta S \tag{6}$$

Where ΔH and ΔS are change in enthalpy and entropy of the system upon ligand binding, respectively, and T is the temperature in Kelvin. Enthalpy is a measure of the total energy of a thermodynamic system, i.e, the sum of the internal energies of the solute and solvent and the amount of energy required to make room for the system (calculated as the product of the system volume and pressure). ΔH is negative in exothermic processes (i.e formations of energetically favorable non-covalent interactions between atoms) and positive in endothermic processes (i.e disruptions of the energetically favorable non-covalent interactions). For a binding process, ΔH reflects the energy change of the system when the ligand binds to the protein. In a non-strict sense, the binding enthalpy is generally treated as the changed in energy resulting from the formations of non-covalent interactions (van der Waals contacts, hydrogen bonds, ion pairs, and any other polar and apolor interactions) at the binding surface. However, the heat effect of a binding reaction is a global property of the entire system, including contributions from both solute and solvent. In reality the net enthalpy change upon binding is a result of forming and disrupting many individual interactions, including the loss of the hydrogen bonds and van der Waals interactions formed between proteins, ligands and the solved, the formation of the non-covalent interactions between the protein and ligand, and the solvent reorganization near the complex surfaces.

Entropy is a measure of how evenly the heat energy will be distributed over the overall thermodynamic system. The second law of thermodynamics determined that the heat always flows spontaneously from regions of higher temperature to regions of lower temperature. This reduces the degree of the order of the initial system, and therefore, entropy could also be viewed as a measure of the disorder or randomness in atoms and molecules in a system. ΔS is a global thermodynamic property of a system, with its positive and negative signs indicating the overall increase and decrease in degrees of the freedom of the system, respectively. The total entropy change associated with binding (the binding entropy ΔS) may be parsed into three entropic terms:

$$\Delta S = \Delta S_{solv} + \Delta S_{conf} + \Delta S_{r/t} \tag{7}$$

Where ΔS_{solv} represents the solvent entropy change arising mainly from surface burial that results in solvent release upon binding, which often makes a favorable contribution to the binding entropy due to its large positive value; ΔS_{conf} represents the conformation entropy change reflecting the changes in the confomration freedom of both the protein and ligand upon binding, which may contribute favorable or unfavorably to the binding entropy because the degree of freedom of the complex may increase or reduce as compared to those of the unbound, free protein and ligand; $\Delta S_{r/t}$ represents the loss of translation and rotational degrees of freedom of the protein and ligand upon complex formation, which reduces the number of particles in solution and contributes unfavorably to the binding entropy. The above three entropic terms determine the net entropy

change, wit positive and negative net entropy change contributing favorably and unfavorably to the binding free energy respectively.

2.1.2 Binding Driving Forces and Enthalpy-Entropy Compensation

2.2 Protein-Ligand Binding Models

Three different models, "lock-and-key", "induced fit", and "conformational selection" have been proposed to explain protein-ligand binding mechanisms.

2.2.1 Diffusion Followed by Collision Is the Prerequisite for Binding

2.2.2 Lock and Key: An Entropy Dominated Process

2.3 Induced Fit

2.3.1 Conformational Selection: A Process in Which Entropy and Enthalpy Play Roles in a Sequential Manner

2.3.2 The Relationship between Lock-and-Key, Induced Fit and Conformational Selection

2.4 Methods Used to Investigate Protein-Ligand Binding Affinity

3 Chem-informatics

3.1 Extended-Connectivity Fingerprints

ECFPs are derived using a variant of the Morgan algorithm, a proposed method for solving the molecular isomorphism problem (when two molecules, with different atom numberings, are the same). The ECFP algorithm makes several changes to the standard Morgan algorithm.

First ECFP generation terminates after a predetermined number of iterations rather than after identifier uniqueness is achieved. The initial atom identifiers and all identifiers after each iteration, are collected into a set; it is this set that defines the extended-connectivity fingerprint. Rather than discarding the intermediate atom identifiers, as in the Morgan algorithm, the ECFP algorithm retains them. Indeed, obtaining these partially disambiguated atom identifiers is the goal of the process. This means that the iteration process does not have to proceed to completion (i.e maximum disambiguation) but is performed for a predetermined number of iterations.

Second,since perfectly accurate disambiguation is not required, algorithmic optimizations are possible. In the standard Morgan process, the identifiers must be carefully recorded after each iteration to avoid mathematical overflow and possible "collision" (where two different atom environments are accidentally mapped to the same identifier). This recoding has the side-effect of creating identifiers that are not comparable between molecules (i.e two identical atom

environments may be given different identifiers to avoid collision on their respective molecules). In the ECFP algorithm, this computationally expensive step is replaced by a fast-hashing scheme. Importantly, the ECFP-hashing scheme generates identifiers that are comparable across molecules.

3.1.1 ECFP Generation Process

The ECFP generation process has three sequential stages:

1. An initial assignment stage in which each atom has an integer identifier assigned to it.

2. An iterative updating stage in which each atom identifier is updated to reflect the identifiers of each atom's neighbors, including identification of whether it is a structural duplication of other features.

3. A duplicate identifier removal stage in which multiple occurrences of the same feature are reduced to a single representative in the final feature list (The occurrence count may be retained if one requires a set of counts rather than a standard binary fingerprint)

The above process is further described as follows. First, atoms are assigned integer identifiers (for example, atoms might use their atomic number). These initial atom identifiers are collected into an initial fingerprint set. Next, each atom collects its own identifier and the identifiers of its immediately neighboring atoms, into an array (the neighbors are ordered using their identifiers, and the order of the attaching bonds, to avoid order-dependence). A hash function is applied to reduce this array back into a new, single-integer identifier. Once all atoms have generated their new identifiers, they replace their old identifiers with their new identifiers. The new atom identifiers are added into the fingerprint set. This iteration is repeated a prespecified number of times. When the specified number of iterations is completed, duplicate identifiers in the set are removed, and the remaining integer identifiers in the fingerprint set define the ECFP fingerprint.

The Morgan algorithm-based updating process is a strictly local operation (that is, each atom collects identifiers only from its immediate neighbors). This results in identifiers that may represent quite large substructures. Given that the process is executed over all atoms in the molecule, the final set of identifiers will contain sub-structural information from all parts of the molecules. Also, since the set is generated by collecting all identifiers up to some number of iterations, the final set contains a mixture of substructures of differing size for each atom in the molecule.

4 Graph Convolutional Networks

4.1 Math Necessary

4.1.1 Normalized Adjacency and Laplacian Matrices

Definition 1 *The **normalized adjacency matrix** is*

$$\mathcal{A} \equiv D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{8}$$

Where A is the adjacency matrix of G and $D = diag(d)$ for $d(i)$ in the degree of node i.

For a graph G (with no isolated vertices), we can see that,

$$D^{-\frac{1}{2}} = \begin{pmatrix} \frac{1}{\sqrt{d(1)}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{d(2)}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sqrt{d(n)}} \end{pmatrix} \tag{9}$$

Definition 2 *The **normalized Laplacian matrix** is*

$$\mathcal{L} \equiv I - \mathcal{A} \tag{10}$$

4.1.2 Chebyshev Polynomials

To do....

4.2 Semi-Supervised Learning Using Gaussian Fields

Suppose there are l labeled points (x_1, y_1), ..., (x_l, y_l), and u unlabeled points x_{l+1}, ..., x_{l+u}; typically $l \ll u$. Let $n = l+u$ be the total number of data points. Consider a connected graph $G = (V, E)$ with nodes V corresponding to the n data points, with nodes $L = \{1, ..., l\}$ corresponding to the labeled points with labels $y_1, ..., y_l$, and nodes $U = \{l+1, ..., l+u\}$ corresponding to the unlabeled points. Our task is to assign labels to nodes U. We assume an $n \times n$ symmetric weight matrix (i.e adjacency matrix) W on the edges of the graph is given. For example, when $x \epsilon \mathcal{R}^{\Updownarrow}$, the weight matrix can be,

$$w_{ij} = exp(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}) \tag{11}$$

Where x_{id} is the d^{th} component of the instance x_i represented as a vector $x_i \epsilon \mathcal{R}^{\Updownarrow}$, and $\sigma_1, ..., \sigma_m$ are length scale hyper-parameters for each dimension. Thus, nearby points in Euclidean space are assigned large edge weight. Other weightings are possible and may be more appropriate when x is discrete or symbolic.

Our strategy is to first compute a real-valued function $f : V \to \mathcal{R}$ on G with certain nice properties, and to then assign labels based on f. We constrain f to take values $f(i) = f_l(i) \equiv y_i$ on the labeled data $i = 1, ..., l$. Intuitively, we want unlabeled points that are nearby in the graph to have similar labels. This motivates the choice of the quadratic energy function,

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij}(f(i) - f(j))^2 \tag{12}$$

To assign a probability distribution on functions f, we form the Gaussian field $p_\beta(f) = \frac{e^{-\beta}E(f)}{Z_\beta}$, where β is an "inverse temperature" parameter, and Z_β is the partition function $Z_B = \int_{f|L=f_l} exp(-\beta E(f))df$, which normalizes over all functions constrained to f_l on the labeled data.

The minimum energy function $argmin_{f|L=f_l} E(f)$ is harmonic; namely it satifsfies $\Delta f = 0$ on unlabeled data points U, and is equal to f_l on the labeled data points L. Here Δ is the combinotorial Laplacian, given in matrix form as $\Delta = D - W$ where $D = diag(d_i)$ is the diagonal matrix with enteries $d_i = \sum_j w_{ij}$ and $W = [w_{ij}]$ is the weight matrix.

The harmonic property means that the value of f at each unlabeled data point is the average of f at neighboring points:

$$f(j) = \frac{1}{d_j} \sum_{i \ j} w_j f(i) \tag{13}$$

for $j = l + 1, ..., l + u$

Which is consistent with our prior notion of smoothness of f with respect to the graph.

Not Done Yet!!!!!!!!!!!!!!!!!!!!!!

4.3 Fast Approximate Convolutions On Graphs

In this section, we provided theoretical motivation for a specific graph-based network model $f(X; A)$ that we will use in the rest of this paper. We consider a multi-layer Graph Convolutional Network (GNC) with the following layer-wise propagation rule:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l W^l) \tag{14}$$

Where $\tilde{A} = A + I_n$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_n is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and W^l is a layer specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function. $H^l \epsilon \mathcal{R}^{NxD}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$

4.4 Spectral Graph Convolutions

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \epsilon \mathcal{R}^N$ (a scalar for every node) with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \epsilon \mathcal{R}^N$ in the Fourier domain, i.e:

$$g_\theta \cdot x = U g_\theta U^T x \tag{15}$$

Where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$, with a diagonal matrix of its eigenvalues Λ and $U^T x$ being the graph Fourier transform of x. We can understand g_θ as a function of the eigenvalues of L, i.e $g_\theta(\Lambda)$. Evaluatin the above equation is compuationaly expensive as mulitplicaton iwth the igenvector matrix U is $\mathcal{O}(N^2)$. Furthermore, computing the eigendecomposition of L in the first palce might be prohibitevley expensive for large graphs. To circumvemnt this problem, it was suggested that $g_\theta(\Lambda)$ can be will-approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ of to K^{th} order:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \tag{16}$$

With a rescaled $\tilde{\Lambda} = \frac{2}{\lambda_{max}} \Lambda - I_N$. λ_{max} denotes the largest eigenvalue of L. $\theta' \epsilon \mathcal{R}^k$ is now a vector of Chebyshev coefficients. The Chebyshev polynomials are recursively defined as $T_k(x) = 2xT_{k-1} - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$.

Going back to our definition of a convolution of a signal x with a filter $g_{\theta'}$, we now have:

$$g_{\theta'} \cdot x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x \tag{17}$$

With $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$; as can be verified by noticing that $(U \Lambda U^T)^k = U \Lambda^k U^T$. Note that this expression is now K-localized since it has a K^{th}-order polynomial in the Laplacian, i.e. it depends only on nodes that are at maximum K steps away from the central node (K^{th}-order neighborhood). The complexity of the above equation is $\mathcal{O}(|\epsilon|)$, i.e. linear in the number of edges. This K-localized convolution can be used to define a convolutional neural network on graphs.

4.5 Layer-Wise Linear Model

4.6 Molecular Graph Convolutions

Here we describe molecular graph convolutions, a deep learning system using a representation of small molecules as undirected graphs of atoms. Graph convolutions extraction meaningful features from simple descriptions of the graph structure-atom and bond properties, and graph distances-to form molecular level representations that can be used in place of fingerprint descriptions in conventional machine learning applications.

4.6.1 Desired Invariants of a Model

A primary goal of designing a deep learning architecture is to restrict the set of functions that can be learned to ones that match the desired properties from the domain. For example, in image understanding, spatial convolutions force the model to learn functions that are invariant to translation. For a deep learning architecture taking a molecular graph as input, some arbitrary choice must be made for the order that the various atoms and bonds are presented to the model. Since that choice is arbitrary, we want:

1. **Property 1** (Order Invariance). *The output of the model should be invariant to the order that the atom and bond information is encoded in the input.*

   We will now gradually construct an architecture which achieves Property 1 while making available a richer space of learnable parameters.

   The first basic unit of representation an *atom layer* which contains an n-dimensional vector associated with each atom. Therefore the atom layer is a 2 dimensional matrix indexed first by atom. The next basic unit of representation is a *pair layer* which contains an n-dimensional vector associated with each pair of atoms. Therefore the pair layer is a 3 dimensional matrix where the first two dimensional are indexed by atoms. Note that the pair input can contain information not just about edges but about any arbitrary pair. Notably, we will encode the graph distance (length of shortest path from one atom to the other) in the input pair layer. the order of the atom indexing for the atom and pair layer inputs must be the same.

   We will describe various operations to compute new atom and pair layers with learnable parameters at every step. Notationally let A^x be the value of a particular atom layer x and P^y be the value of a particular pair layer y. A^x_a refers to the value of atom a in atom layer x and $P^y_{(a,b)}$ refers to the value of pair (a, b) in pair layer y. In order to achieve Property 1 for the overall architecture, we need a different type of invariance for each atom and pair layer.

2. **Property 2** (Atom and pair permutation invariance). *The values of an atom layer and pair permute with the original input layer order. More precisely, if the inputs are permuted with a permutation operator Q, then for ll layers x, y, A^x and P^y are permuted with operator Q as well*

   In other words, Property 2 means that from a single atom's (or pair's) perspective, its value in every layer is invariant to the order of the other atoms (or pairs).

   Since molecules are undirected graphs, we will also maintain the following:

3. **Property 3** (Pair order invariance). *For all pair layers y, $P^y_{(a,b)} = P^y_{(b,a)}$*

Property 3 is easy to achieve at the input layer and the operations below will maintain this. Properties 2 and 3 make it easy to construct a molecule-level representation from an atom or pair such that the molecule-level representation achieves Property 1.

4.6.2 Invariant-preserving operations

We now define a series of operations that maintain the above properties.

Throughout, f represents an arbitrary function and g represents and arbitrary *commutative* function. In this work, f is a learned linear operator with a ReLU activation function and g is a sum.

The most trivial operation is to combine one or more layers of the same type by applying the same operation to every atom or pair. Precisely, this means if you have layers $x_1, ..., x_n$ and function f, you can compute a new atom layer from the previous atom layer ($A \to A$) as,

$$A_a^y = f(A_a^{x_1}, A_a^{x_2}, ..., A_a^{x_n}) \tag{18}$$

or pair layer from the previous pair layer ($P \to P$) as,

$$P_{(a,b)}^y = f(P_{(a,b)}^{x_1}, P_{(a,b)}^{x_2}, ..., P_{(a,b)}^{x_n}) \tag{19}$$

Since we apply the same function for every atom/pair, we refer to this as a convolution. All the transformation we develop below will have this convolutional nature of applying the same operation to every atom/pair, maintaining Property 2.

Next, consider an operation that takes a pair layer x and constructs an atom layer y ($P \to A$). Formally,

$$A_a^y = g(f(P_{(a,b)}^x), f(P_{(a,c)}^x), f(P_{(a,d)}^x), ...) \tag{20}$$

In other words, take all pairs of which a is a part, run them through f, and combine them with g. Note that Property 3 means we can choose an arbitrary one of $P_{(a,b)}^x$ or $P_{b,a}^x$.

The most interesting construction is making a pair layer from an atom layer ($A \to P$). Formally,

$$P_{ab}^y = g(f(A_a^x, A_b^x), f(A_b^x, A_a^x)) \tag{21}$$

Note that just applying g to A_a^x and A_b^x would maintain Properties 2 and 3 but we use this more complex form. While commutative operators (such as max pooling) are common in neural networks, commutative operators *with learnable parameters* are not common. Therefore, we use f to give learnable parameters while maintaining the desired properties.

Once we have all the primitive operations on atom and pair layers ($A \to A, P \to P, A \to P, P \to A$), we can combine these into one module. We call this the Weave module because the atoms and pair layers cross back and forth to each other. The model can be stacked to an arbitrary depth similar to the Inception module that inspired it.

4.6.3 Molecule-level features

The construction of the Weave module maintains Properties 2 and 3. What about the overall order invariance (Property 1)? At the end of a stack of Weave modules we are left with an n-dimensional vector associated with every atom and an m-dimensional atom associated with every pair. *We need to turn this into a molecule-level representation with some* ***commutative function*** *of these vectors.*

In related work, a simple unweighted sum is often used to combine order dependent atom features into order-independent molecule-level features. However, reduction to a single value does not capture the distribution of learned features. We experimented with an alternative approach and created "fuzzy" histograms for each dimension of the feature vector.

A fuzzy histogram is described by a set of *membership functions* that are functions with range $[0, 1]$ representing the membership of the point in each histogram bin. A standard histogram has membership functions which are 1 in the bin and 0 everywhere else. For each point, we normalize so that the total contribution to all bins is 1. The value of a bin in the histogram over all points is just the sum of the normalized contributions for all the points. A histogram is constructed for each dimension of the feature vectors and the concatenation of those histograms is the molecule-level representations.

Before the molecule-level featurization, we do one final convolution on the atoms. Since molecule-level featurization can be a major bottleneck in the model, this convolution expands the depth so that each dimension of the atom feature vector contains less information and therefore less information is lost during the molecule-level-featurization. On this convolution, we do not use a ReLU activation function to avoid the histogram having many points at zero. Once you have a molecular-level representation, this becomes a more standard multitask problem.

In models with multiple Weave modules it is conceivable to vary the convolution depths in a module specific way.

5 Molecular File Formats: PDB and MOL2

6 Methods

he proposed model is a multi-branch graph neural network designed for regression tasks, specifically to predict binding affinity in terms of $logK_d/logK_i$ values. The architecture simultaneously processes molecular graph representations of proteins and ligands, along with complementary tabular features, to produce a continuous scalar output. Each of the two graph-based input branches—one for the protein and one for the ligand—consists of a series of graph convolutional layers (GCNConv) interleaved with ReLU activation functions. These layers operate on 30-dimensional atom-level node features that are first standardized via batch normalization. In the best-performing configuration, each branch em-

ployed a two-layer GCN encoder with hidden dimensions of 8 units per layer, balancing representational capacity with model simplicity.

Following graph convolution, node-level embeddings are aggregated into fixed-size graph-level representations using global mean pooling, producing one embedding vector per graph instance in a batch. These pooled protein and ligand representations are then concatenated with a 7-dimensional vector of scalar molecular descriptors, including features such as molecular weight, partial charge extremes, and topological polar surface area. This fused representation, integrating structural and physicochemical information, is passed through a fully connected linear layer to produce a single continuous output representing the predicted $logK_d/logK_i$ value. This design proved effective for structure-based affinity prediction, leveraging both learned graph-based features and expert-derived molecular descriptors.

Node features extracted for each protein and ligand are provided in the table below.

Node Feature	**Description**	**Size**
Atom type	C, N, O, F, P, S, Cl, Br, I, other atoms (one-hot vector)	10
Formal charge	Integer electronic charge	1
Hybridization	sp, sp^2, sp^3 (one-hot vector)	3
Hydrogen bonding	Hydrogen bond donor or acceptor (one-hot vector)	2
Aromatic	Whether this atom is part of an aromatic system (one-hot vector)	1
Degree	Atom degree (0–5) (one-hot vector)	6
Number of Hydrogens	Hydrogens (0–4) attached to this atom (one-hot vector)	5
Chirality	R or S configuration (one-hot vector)	2
Partial Charge	Calculated partial charge	1
Total		**30**

Table 1: Atom-level features used in the graph representation.

Edge features extracted for each protein and ligand are provided in the table below.

Edge Feature	Description	Size
Bond type	single, double, triple, or aromatic (one-hot vector)	4
Same ring	Whether atoms are in the pair are in the same ring (one-hot vector)	1
Conjugated	Is the bond conjugated (one-hot vector)	1
Stereo	Stereo configuration of the bond (one-hot vector)	2
Total		11

Molecular properties extracted for each protein and ligand are provided in the table below.

Molecular Property	**Description**	**Size**
Exact Molecular Weight	The precise molecular mass calculated using the most abundant isotope of each element.	1
Number of Radical Electrons	The number of unpaired electrons in the molecule, indicating reactive species.	1
Number of Valence Electrons	The total number of valence electrons available for bonding across the molecule.	1
Maximum Partial Charge	The highest partial atomic charge within the molecule, indicating electron-rich sites.	1
Minimum Partial Charge	The lowest partial atomic charge within the molecule, indicating electron-poor sites.	1
Number of H Donors	The number of hydrogen bond donors (typically -OH or -NH groups).	1
TPSA	Topological polar surface area, a measure of the surface area occupied by polar atoms.	1
Total		**7**

Table 2: Tabular molecular descriptors used in the model.

The best-performing model utilized a graph convolutional neural network architecture with 30-dimensional atom-level node features and a two-layer GCN encoder with hidden dimensions of 8 units each for both the protein and ligand graph streams. Tabular molecular features, comprising 7 scalar descriptors, were concatenated with the graph-level embeddings following global pooling. This fused representation was passed through a fully connected linear layer to predict continuous log K¡sub¿D¡/sub¿ values. The model achieved optimal performance with this lightweight configuration, effectively integrating structural and physicochemical information from all three input modalities.

7 Results

The full dateset was split into training and testing subsets using an 80/20 split. The model was trained for 250 epochs using the Adam optimizer and mean squared error (MSE) loss function, with a learning rate of 0.001. The training process took a total of 176.00 seconds. Figure 1 shows the training and validation loss curves over the course of training, which demonstrate semi-stable convergence up to approximately 125 epochs, followed by signs of overfitting. Despite this, the model retained some generalization ability on the validation set.

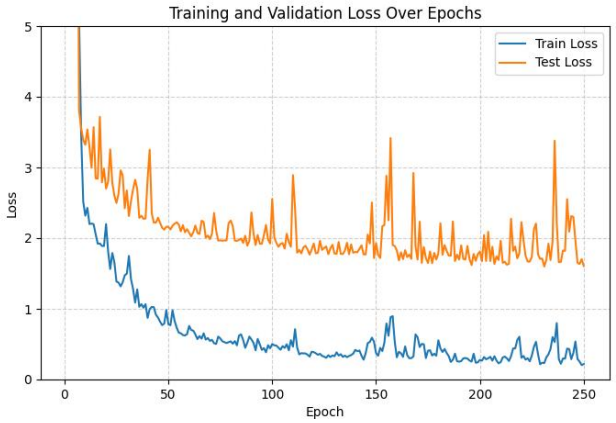

Figure 1: Training and validation loss

The model achieved a root mean squared error (RMSE) of 1.254 on the validation set, corresponding to a relative uncertainty of 14.8%.

8 Discussion

The results presented in this work highlight the potential of graph neural network (GNN) architectures to effectively capture molecular information relevant to predicting key physical properties such as binding affinity. Specifically, this study demonstrates that GNNs can extract meaningful structural features from both protein and ligand graphs, showing promise for broader application in molecular property prediction tasks.

Moreover, the multi-modal framework employed here—integrating graph-based representations with scalar molecular descriptors—further underscores the value of combining heterogeneous data sources. This fusion of structural and physicochemical features allowed the model to leverage complementary information and improve predictive performance.

Importantly, this approach deviates from conventional practices that typically rely solely on protein features. By incorporating detailed structural information from both the protein and the ligand, the proposed method achieved strong performance in predicting $logK_d/logK_i$ values, suggesting that jointly modeling protein-ligand interactions is a more informative and effective strategy. This insight supports future development of multi-modal, graph-based models for structure-based drug discovery and affinity prediction.

9 References

1. Du, Xing, et al. "Insights into Protein–Ligand Interactions: Mechanisms, Models, and Methods." MDPI, Multidisciplinary Digital Publishing Institute, 26 Jan. 2016, www.mdpi.com/1422-0067/17/2/144.

2. M;, Rogers D;Hahn. "Extended-Connectivity Fingerprints." Journal of Chemical Information and Modeling, U.S. National Library of Medicine, pubmed.ncbi.nlm.nih.gov/20426451/. Accessed 9 May 2025.

3. Kipf, Thomas N., and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." arXiv.Org, 22 Feb. 2017, arxiv.org/abs/1609.02907.

4. Kearnes, Steven, et al. "Molecular Graph Convolutions: Moving beyond Fingerprints." arXiv.Org, 18 Aug. 2016, arxiv.org/abs/1603.00856.