CAN-GAT: Drug Target Interaction Prediction for Molecular Cancer Drugs Employing Graph Attention Networks

Albert Guo, Adarsh Padalia, Yunzhe Qiao, Mark Wang

Wednesday, March 27, 2025

Abstract

Cancer is currently the leading cause of death in Canada [9], and a leading cause of death worldwide [10]. Despite its prominence, cancer remains difficult to treat in part due to the similarity between cancer cells and healthy cells. Targeted molecular therapies have shown promise, designed to be able to bind specifically to proteins that cancers use to signal and replicate. The first targeted cancer drug, imatinib, was approved in 2001 and has shown significant efficacy against chronic myelogenous leukemia (CML) [1]. Since then, other targeted therapies, such as those for treating epidermal growth factor receptor positive non-small cell lung carcinoma (EGFR+ NSCLC) [3], have been developed. Many of these therapies are highly effective; however, substantial barriers to the discovery of these targeted therapies remain. In particular, predicting specific drug-target interactions is challenging due to the complex nature of these interactions, which depend on the amino acid sequence of the protein, its folded structure, and the molecular structure of the drug candidate. In this paper, we analyze the use of a novel Graph Attention Network (GAT) [6] for assessing drug-target interaction predictions on the CandidateDrug4Cancer dataset [11]. We were able to achieve significant improvements over the existing DTI prediction models, such as those employing Morgan fingerprints [2] combined with XGBoost [4], and we believe that our model shows significant promise for the complex field of drug discovery.

Introduction

Drug-Target Interaction (DTI) prediction has long been recognized as a critical step in the drug development pipeline [5]. Traditionally, this process relied heavily on experimental screening, which was both time-consuming and costly. In recent years, advances in artificial intelligence (AI) have introduced new approaches for accelerating drug discovery. Among these, Graph Neural Networks (GNNs) have emerged as a powerful tool capable of leveraging the structural information inherent in molecular graphs [6].

In this study, we focused on the CandidateDrug4Cancer dataset [11], which included 29 cancer-associated protein targets, 54,869 drug molecules, and 73,770 known drug-protein interactions. Each molecule was represented as a graph, where nodes corresponded to atoms and edges represented chemical bonds. Our primary task was to predict the likelihood of a biological interaction between a given drug and a cancer target protein based on this structural data.

We implemented a Graph Attention Network (GAT) using PyTorch [6] to perform DTI prediction and compared its performance to traditional baseline methods, such as Morgan fingerprints [2] combined with XGBoost [4]. We also visualized the model's output using Feature Activation Maps [7] to interpret the predicted efficacy of specific drugs. The objective was to assess whether GATs could offer improved accuracy and predictive insight over conventional approaches in the context of cancer-targeted drug screening.

Computational Overview

To implement this, we divide the computing process into the following steps: Data Processing, Graph Construction, Neural Network Model Architecture, Training Process, Evaluation, and Hyperparameter Tuning and Optimization.

- **Data Processing**

  1. Dataset Structure

  Typically, each row in the interaction table provides the following columns:

Column	Type	Description
`ChEMBL_ID`	String	ChEMBL identifier of the compound (e.g., `CHEMBL123456`).
`Target_ID`	String	ChEMBL identifier of the target (e.g., `CHEMBL1824`).
`pChEMBL_Value`	Float	$-\log_{10}$ of the measured IC_{50}.
`SMILES`	String	Chemical structure in SMILES format.
`Protein`	String	Full amino acid sequence of the target protein.
`Label`	Boolean	Binary indicator of active/inactive

2. Equations for Label Computation and Molecular Features

2.1 pChEMBL and Activity Label

The dataset leverages a pChEMBL value, defined by:

$$p\text{ChEMBL} \;=\; -\log_{10}\big(\text{IC}_{50}\text{ in molar}\big),$$

which transforms IC_{50} from molar units into a logarithmic scale.

A threshold $\texttt{pChEMBL} \geq 7.0$ (corresponding to $IC_{50} \leq 100$ nM) denotes a *potent* interaction. Hence, we define the activity label as:

$$\text{Activity\_Label} \;=\; \begin{cases} 1, & \text{if } p\text{ChEMBL} \geq 7.0, \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Molecular Graph Features

For each molecule, we parse the SMILES string into a graph. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of atoms, and $E = \{(v_i, v_j) \mid \text{bond exists between } v_i \text{ and } v_j\}$ the set of edges.

Each atom node v_i has features such as:

1. *Atom Type* (atomic number)
2. *Formal Charge*
3. *Degree* (number of covalent bonds to v_i)
4. *Hybridization* (e.g., sp, sp2, sp3)
5. *Aromaticity*

Each bond (v_i, v_j) has features like:

1. *Bond Type* (Single, Double, Triple, Aromatic)
2. *Conjugation*
3. *Ring Membership*

3. Data Preprocessing and Splitting

Step 1: Loading Data. We first read the CSV file that contains the compound–target interactions:

```
import pandas as pd

df = pd.read_csv("CandidateDrug4Cancer_full.csv")
print(df.head())
```

Step 2: Creating Activity Labels. If the label is not already present, we define:

```
df["Activity_Label"] = (df["pChEMBL_Value"] >= 7.0).astype(int)
```

Step 3: Splitting into Train/Validation/Test. We split randomly or by target:

```
from sklearn.model_selection import train_test_split

train_df, temp_df = train_test_split(df, test_size=0.3, stratify=df["Activity_Label"])
val_df, test_df = train_test_split(temp_df, test_size=0.5, stratify=temp_df["
    Activity_Label"])
```

4. Graph Construction (Feature Computation)

Using RDKit, we convert each SMILES to a molecule and extract the heavy-atom graph:

```python
from rdkit import Chem
from typing import Any

def construct_molecular_graph(smiles_str: str) -> dict[str, list[Any]]:
    mol = Chem.RemoveHs(Chem.MolFromSmiles(smiles_str)) # remove explicit H atoms

    node_features = []
    adjacency_list = []
    edge_features = []

    for atom in mol.GetAtoms():
        feats = {
            "atomic_num": atom.GetAtomicNum(),
            "formal_charge": atom.GetFormalCharge(),
            "degree": atom.GetDegree(),
            "hybridization": str(atom.GetHybridization()),
            "aromatic": int(atom.GetIsAromatic())
        }
        node_features.append(feats)

    for bond in mol.GetBonds():
        i = bond.GetBeginAtomIdx()
        j = bond.GetEndAtomIdx()
        bond_feat = {
            "bond_type": str(bond.GetBondType()),
            "conjugated": int(bond.GetIsConjugated()),
            "ring": int(bond.IsInRing())
        }
        edge_features.append(((i,j), bond_feat))
        % Undirected adjacency
        adjacency_list.append((i, j))
        adjacency_list.append((j, i))

    return {
        "node_features": node_features,
        "edge_features": edge_features,
        "adjacency_list": adjacency_list
    }
```

Explanation: We record atom-level descriptors (atomic number, formal charge, etc.) and bond-level descriptors (bond type, conjugation, ring membership). The adjacency list is built from each bond pair (i, j). This structure can be fed into a graph neural network or converted into other GNN frameworks like PyTorch Geometric or DGL.

- **Construct Graph Attention Network (GAT)**: The Graph Attention Network architecture first processes each drug's molecular graph through multiple GAT layers. Within each GAT layer, a node updates its embedding by computing attention scores for its neighbors. These scores highlight the relative importance of different neighboring nodes, and multi-head attention further stabilizes learning by combining insights from multiple parallel attention mechanisms [6]. After the message passing phase, a pooling operation generates a single embedding for the entire molecule, typically by taking a mean or weighted sum of node embeddings. The final output of the GAT model is the predicted pChEMBL for the drug-protein pair.

Remark: Graph Attention Network Equations

- **Attention Coefficients:** For each node i, we compute attention scores a_{ij} toward each neighbor j as follows:

$$\alpha_{ij} = \frac{\exp\Big(\text{LeakyReLU}\big(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]\big)\Big)}{\sum_{k\in\mathcal{N}(i)} \exp\Big(\text{LeakyReLU}\big(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_k]\big)\Big)},$$

where $\mathbf{h}_i$ is the current feature vector for node i, $\mathbf{W}$ is a trainable weight matrix, and $\mathbf{a}$ is a learnable vector for the attention mechanism [6].

- **Node Representation Update:** After normalizing coefficients, we update the node representation:

$$\mathbf{h}_i' = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W} \mathbf{h}_j\Big),$$

where σ is a non-linear activation function (ReLU in our case) [6].

- **Training Process**: Training was preformed on an Nvidia A5000 (Ampere) GPU. During training, the model uses Huber loss to predict whether a drug-protein pair is likely to bind. The Adam optimizer updates the network parameters, and regularization methods, such as dropout and weight decay, help the network generalize better. Early stopping is employed by monitoring the validation loss and halting training when performance ceases to improve, which avoids excessive overfitting [5].

- **Evaluation**

1. Our study compared the GAT-based approach against a baseline model that uses Morgan fingerprints [2] to encode drug molecules. In the baseline method, circular fingerprints capture substructures around each atom, and a gradient boosting algorithm, XGBoost [4], processes these representations together with simpler protein features. By evaluating the two approaches side by side, we assess whether the structural awareness introduced by graph neural networks provides a significant performance boost over traditional fingerprint-based techniques.
2. Model performance is evaluated using multiple metrics to capture different aspects of prediction quality. Since this is a prediction algorithm and not a classification algorithm, we define the prediction as correct when:

$$|\textit{predicted ChEMBL score} - \textit{actual pChEMBL score}| < 1.0$$

and define the accuracy as:

$$\frac{\textit{number of correct predictions}}{\textit{number of samples}}.$$

Additionally, we evaluate the model in terms of mean absolute error (MAE) and mean squared error (MSE) [5].

An object-oriented Graph class is implemented to simplify the process of storing and retrieving molecular structures. This class keeps track of adjacency lists, node features, and bond attributes, and offers methods for adding nodes, adding edges, and retrieving neighbor information [5].

In addition, the `GATModel` class encapsulates the GAT layers, along with a residual connection and a fully connected layer for the final prediction. It provides a `forward(drug_graph, protein_embedding)` method returning an interaction probability. A `Trainer` class is also implemented to coordinate the training loop, including batch processing, loss computation, backpropagation, and metric logging [5].

- **Hyperparameter Tuning**: Hyperparameter tuning explores a range of values for the learning rate, hidden dimension, attention heads, and dropout rates. A systematic approach, such as grid search or Bayesian search, helps identify settings that maximize validation performance while preserving generalization [12]. Additional experiments can examine how the number of GAT layers affects the model's ability to capture higher-order structural relationships within each molecule.

- **Visualizing the Results**: Upon training a successful model, we can determine the specific nodes and edges of the drug molecule that improve or impair the drug's efficacy using Feature Activation Maps [7]. For example, given a drug/protein pair, we construct a graphical representation of the drug molecule (using a library like `networkx`) that utilizes color to show the relative contribution of different atoms and bonds to the efficacy of the drug [7].

Listing 1: Python implementation of a Graph class for molecular data

```
class Graph:
    """
    A Graph class to represent a molecular graph with atom-level and bond-level features.

    Instance Attributes:
        - neighbors: A dictionary mapping node IDs to a list of neighboring node IDs.
        - node_features: A dictionary mapping node IDs to a dictionary of that node's features.
        - edge_features: A dictionary mapping (node_i, node_j) to a dictionary of bond-level features.
    """
```

Listing 2: Scratch implementation of `GATModel` and `Trainer` classes

```
class GATModel:
    """
    A model class containing the multi-head GAT architecture and downstream fully connected layers for
        final prediction.

    Instance Attributes:
        - num_heads: Number of attention heads.
        - hidden_dim: Dimensionality of the hidden layers.
        - gat_layers: A collection of GAT layers.
        - fc_layers: Fully connected layers to combine graph embeddings with protein features for final
            prediction.
    """


class Trainer:
    """
    A class used for coordinating the training loop, including data loading, backpropagation, and logging.

    Instance Attributes:
        - model: The GAT-based model to be trained.
        - optimizer: The optimizer used for parameter updates.
        - criterion: The loss function for computing training loss.
        - metrics: A collection of metric functions for evaluation.
        - device: The device (CPU or GPU) on which training is performed.
    """
```

Running the Model

Ensure you have `python 3.13` installed. To run the model, from the base directory, run:
`pip install -r requirements.txt`
`python main.py`

Note that `tkinter` has known stability and compatibility issues on macOS, as the Apple-supplied python installation does not work well with `tkinter`. If you are using an Apple Silicon device, you may need to reinstall Python from the official Python website or attempt to run the program on a different device.

Also know that PythonTA has issues running in some files. This is likely due to a bug within PythonTA, and how it interacts with other modules.

Results

Our model was able to significantly outperform the baseline Morgan Fingerprint technique [2], [4], and was able to accurately predict the pChEMBL scores of several drugs. On the testing dataset, we achieved achieving a MAE of 0.540, and a ± 1 accuracy of 0.844. On the testing dataset, we achieved ± 1 accuracy of 84.4%, meaning that 84.4% of predictions by the model were within ± 1 of the actual pChEMBL score. This indicates that our model is able to accurately predict the activity of the majority of molecular cancer drugs. In addition, our model was able to achieve an AUC of 0.901, which is extremely high.

Model	± 1 Accuracy	MSE	MAE	R^2	AUC
CAN-GAT	**0.844**	**0.540**	**0.551**	**0.645**	**0.901**
Morgan Fingerprints + XGBoost	0.741	0.779	0.707	0.484	0.746

Table 1: Performance metrics for different models

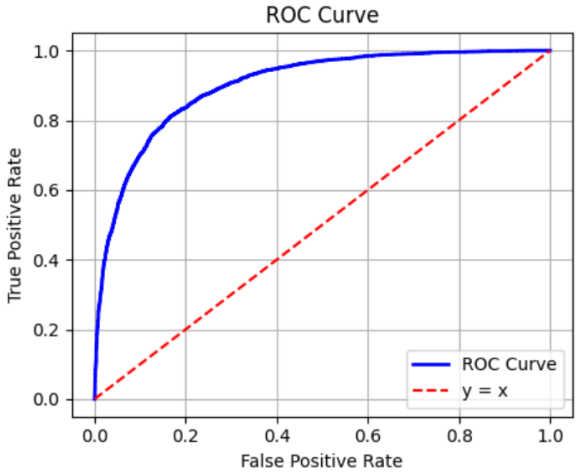

Figure 1: AUC curve of model on testing data.

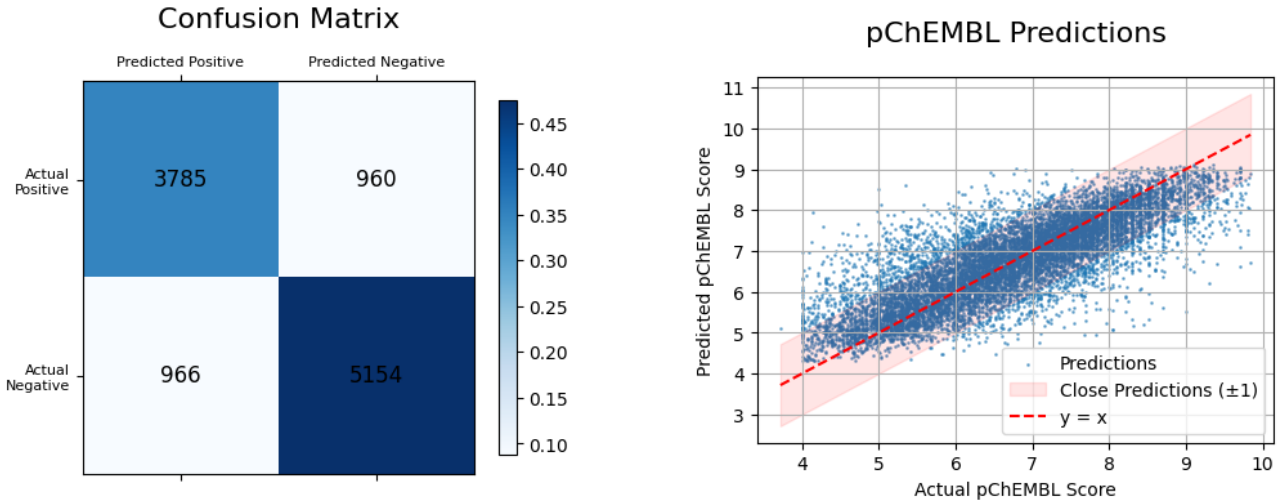

Figure 2: Confusion Matrix and scatterplot of model predictions on testing data.

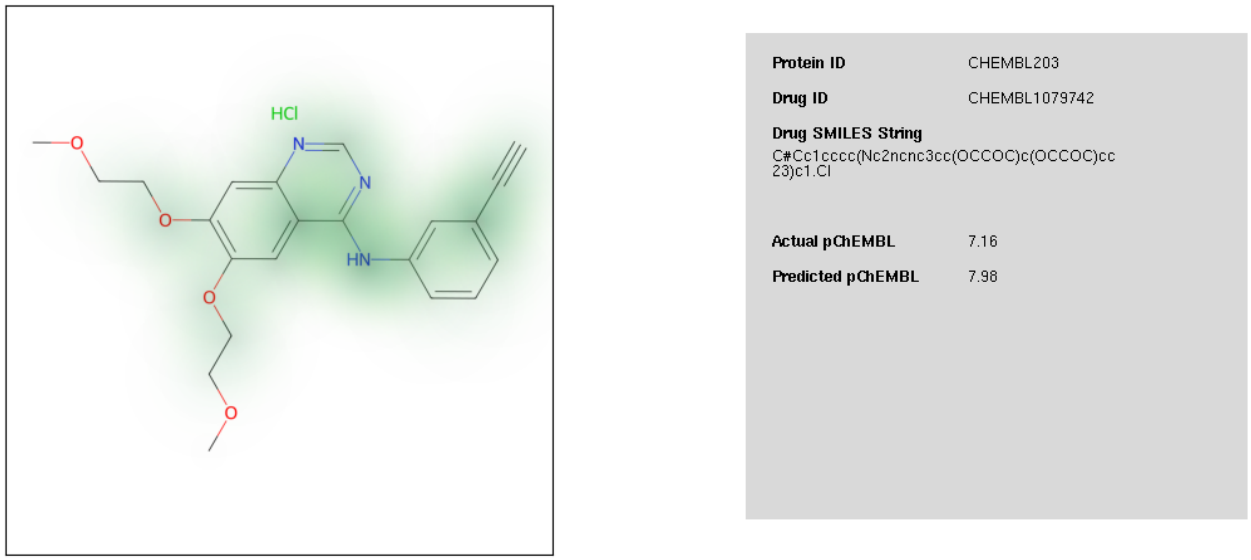

Figure 3: Our model's prediction of Erlotinib's (Tarceva) activity, a commonly used epidermal growth factor receptor (EGFR) inhibitor used in treating non-small cell lung carcinoma (NSCLC) and pancreatic adenocarcinoma (PDAC) [3]. Green areas positively contribute to the predicted score; red areas negatively contribute to the predicted score.

Discussion

Our CAN-GAT model demonstrated promising performance in predicting drug-target interactions on the CandidateDrug4Cancer dataset [11]. Compared to the baseline approach using Morgan fingerprints with XGBoost [2], [4], CAN-GAT achieved lower MAE and MSE, and higher ± 1 accuracy, suggesting improved discrimination between active and inactive drug-target pairs. When benchmarked against other graph-based DTI prediction models, CAN-GAT aligns closely with approaches like those proposed by Veličković et al. [6]. Unlike traditional cheminformatics methods that rely on fixed molecular descriptors, CAN-GAT leverages the full graph structure, offering a more flexible representation of chemical interactions. We aim to further refine our model and test against other state-of-the-art approaches such as the one proposed by Tsubaki et al. [8].

Visualization of attention weights through Feature Activation Maps [7] provided valuable insights into the molecular basis of drug efficacy. Generally speaking, amine functional groups and rings had high activation scores, indicating that they were especially important in determining the activity of compounds. These findings align with known principles of medicinal chemistry [5] and could help to make our model more interpretable, rather than simply being a "black box" prediction model.

Despite its strengths, CAN-GAT faces several limitations. The CandidateDrug4Cancer dataset, while comprehensive, includes only 29 cancer-associated protein targets, potentially limiting the model's ability to generalize across diverse protein families [11]. Additionally, the computational complexity of multi-head attention mechanisms increases training time compared to simpler baselines, posing a challenge for scaling to larger datasets [12]. Although for this experiment, it was possible to train on a single GPU, further optimizations of the model could increase training and inference speed. Generalization to novel drugs or proteins outside the training distribution also remains an open question.

Future work could enhance CAN-GAT by integrating 3D protein structure data, such as residue contact maps, to better model drug-target interfaces. Scaling the approach to larger, more diverse datasets—potentially through transfer learning [13]—could improve robustness. Finally, collaborating with experimentalists to validate top-ranked predictions in vitro would bridge the gap between computational predictions and real-world drug discovery, amplifying the practical impact of this approach. Furthermore, it may be possible to engineer compounds that are specifically active against a protein by systematically adding and removing functional groups in order to maximize the predicted score. This would vastly accelerate the speed of drug discovery.

Conclusion

Overall, our approach of applying a novel GAT architecture in predicting protein-molecule interactions shows promise. We were able to outperform the baseline model and achieve high evaluation metrics. Future work will focus on applying the model to other datasets, refining the architecture of the model further, and using the model to generate active compounds that can be tested in real-world environments.

Acknowledgments

The authors would like to acknowledge the Toronto Computational Imaging Group for providing the computational resources needed for this study.

References

[1] B. J. Druker, M. Talpaz, D. J. Resta, *et al.*, "Efficacy and safety of a specific inhibitor of the BCR-ABL tyrosine kinase in chronic myeloid leukemia," *New England Journal of Medicine*, vol. 344, no. 14, pp. 1031–1037, 2001. DOI: 10.1056/NEJM200104053441401.

[2] D. Rogers and M. Hahn, "Extended-Connectivity Fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, 2010. DOI: 10.1021/ci100050t.

[3] R. Rosell, E. Carcereny, R. Gervais, *et al.*, "Erlotinib versus standard chemotherapy as first-line treatment for European patients with advanced EGFR mutation-positive non-small-cell lung cancer (EURTAC): a multicentre, open-label, randomised phase 3 trial," *The Lancet Oncology*, vol. 13, no. 3, pp. 239–246, 2012. DOI: 10.1016/S1470-2045(11)70393-X.

[4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[5] X. Chen, C. C. Yan, X. Zhang, *et al.*, "Drug–target interaction prediction: databases, web servers and computational models," *Briefings in Bioinformatics*, vol. 17, no. 4, pp. 696–712, 2016. DOI: 10.1093/bib/bbv066.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[7] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability Methods for Graph Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 772–10 781. DOI: 10.1109/CVPR.2019.01102.

[8] M. Tsubaki, K. Tomii, and J. Sese, "Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences," *Bioinformatics*, vol. 35, no. 2, pp. 309–318, 2019. DOI: 10.1093/bioinformatics/bty535.

[9] Canadian Cancer Society, Statistics Canada and Public Health Agency of Canada, "Canadian Cancer Statistics 2021," Canadian Cancer Society, Toronto, Canada, 2021, Released Nov 2021. Available from: https://cancer.ca/en/research/cancer-statistics/canadian-cancer-statistics-2021.

[10] H. Sung, J. Ferlay, R. L. Siegel, *et al.*, "Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries," *CA: A Cancer Journal for Clinicians*, vol. 71, no. 3, pp. 209–249, 2021. DOI: 10.3322/caac.21660.

[11] X. Ye, Z. Li, F. Ma, *et al.*, "CandidateDrug4Cancer: An Open Molecular Graph Learning Benchmark on Drug Discovery for Cancer," *arXiv preprint arXiv:2203.00836*, 2022.

[12] G. C. Kanakala, R. Aggarwal, D. Nayar, and U. D. Priyakumar, "Latent Biases in Machine Learning Models for Predicting Binding Affinities Using Popular Data Sets," *ACS Omega*, vol. 8, no. 3, pp. 2389–2397, 2023. DOI: 10.1021/acsomega.2c06246.

[13] X. Ouyang, Y. Feng, C. Cui, Y. Li, L. Zhang, and H. Wang, "Improving generalizability of drug–target binding prediction by pre-trained multi-view molecular representations," *Bioinformatics*, vol. 41, no. 1, btaf002, 2024. DOI: 10.1093/bioinformatics/btaf002.