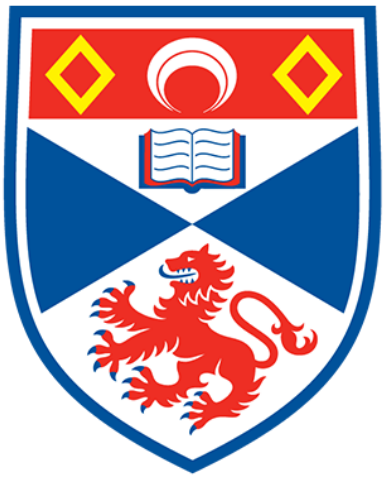
Exploring the Effectiveness of Graph Neural Networks in Predicting Candidate Drugs and Protein-Ligand Binding



University of
St Andrews

Ayam Babu

ab550@st-andrews.ac.uk

Supervised by: Dr. John Mitchell

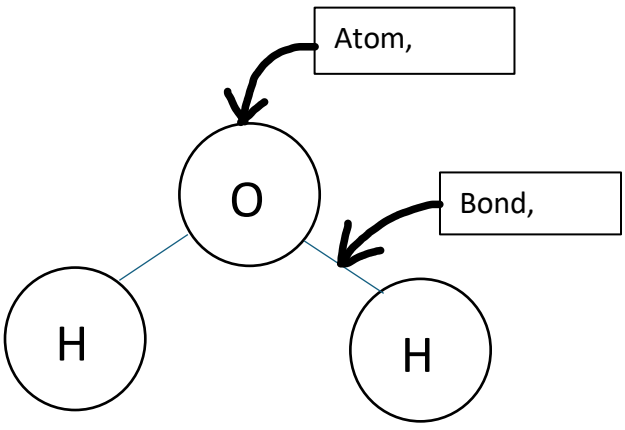
July 2024

Introduction:

It costs around $1 Billion (Wouters, et al., 2020) and takes 10-15 years (Cancer Research UK, 2022) to bring a new drug to market. This process has become more expensive and time-consuming, in a phenomenon known as Eroom's Law (Scannell, et al., 2012). The integration of Artificial Intelligence into different stages of the drug discovery pipeline can potentially reverse this trend. One of these stages is virtual screening, which involves evaluating candidate drug molecules to determine their potential to treat diseases.

Highlighting potential drug molecules helps scientists focus on what molecules they should investigate. The goal is to find drug molecules that can bind with the target protein. Genetic diseases often lead to the production of proteins that can't perform their usual functions. These produced proteins need to be targeted by the drug molecules, thus their name "target protein".

The target proteins and candidate drug molecules can be represented as graphs, where vertices represent atoms and the edges represent bonds. For further clarification, see the diagram below:



Example of a water molecule represented as a graph

These graphs can be analyzed through Graph Neural Networks (GNNs), a type of AI model that processes graphs. This paper evaluates the effectiveness of GNNs in identifying what drug molecules can treat diseases.

Literature Review:

There have been multiple computational approaches to virtual screening, including knowledge-based and other GNN-based methods.

Knowledge-based methods define specific methods to determine the likelihood of a candidate drug molecule and protein would bind. DLIGAND2, for example, uses chemical knowledge to determine the energy between atoms in the target protein and candidate drug molecules (Chen, et al., 2019). This approach relies on known chemical principles, rather than knowledge which is machine-learned.

Though this project's method uses GNNs, their usage in virtual screening is not novel. Models like Neural Fingerprinting (Gonczarek, et al., 2018) and ParaVS-ND (Wu, et al., 2021) both employ Graph Neural Networks. Neural Fingerprinting model explicitly defines a Neural Fingerprinting layer, which uses a Graph Neural Network to turn a protein's binding site, where a drug molecule could bind, or candidate drug molecule into a fingerprint, then processes this fingerprint with a Neural Network to predict binding likelihood.

ParaVS-ND also uses GNNs to process protein pockets and candidate drug molecules, which are then fed into a classifier. Their GNNs are partly inspired by natural language models, like those used in GPT.

This project utilizes a standard Graph Convolutional Network to thoroughly analyze the protein and candidate drug molecule graphs. Most of these models focus on protein pockets, but this project's method would be using the whole protein. It would be interesting to observe the differences and compare them together.

A challenge with GNN-based solutions is their "black box" nature, making it difficult to understand the reasoning behind the methods' results. This contrasts with a knowledge-based method like DLIGAND2, where the candidate drug and protein binding potential are

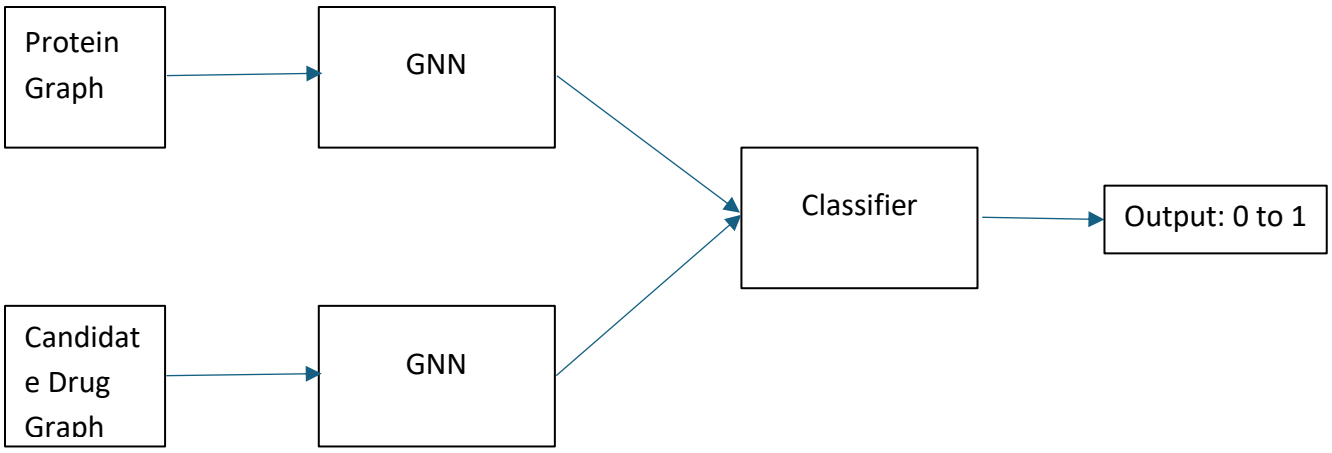
explicitly calculated. Comparing how Graph Neural Networks perform against knowledge-based methods will provide valuable insights.

Methodology:

Neural networks process different types of data, such as text, images, graphs, and tables. The data enters through the input layer, then it is passed through hidden layers, which transform the input to make predicting the correct output easier. The data transformations through the hidden layers are known as embeddings because they simplify complex data. Finally, the output layer makes predictions from the embeddings fed in. These predictions can be numerical (like the price of a house, known as regression problems) or categorical (like red or green, known as classification problems).

Predicting whether a drug molecule can bind to a protein is a classification problem. The inputs are two graphs: one representing the target protein and the other representing the candidate drug molecule. Due to differences between target proteins and drug molecules, two separate Graph Neural Networks are used to focus on different components of their graphs. These GNNs do not have output layers by themselves because to predict binding potential, both graphs need to be considered together. Instead, the transformed outputs from both GNNs are fed into another neural network with an output layer. This way, the Graph Neural Networks effectively simplify the protein and candidate drug molecule graphs for classification, while preserving essential insights.

In the final neural network, the graph embeddings are further transformed for accurate predictions. The output layer outputs a number between 0 and 1, indicating the confidence that the candidate drug molecule will bind with the target protein. The following diagram illustrates this process:



For training the model, a weighted Binary Cross Entropy loss function was used, as shown below:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(\beta_p \cdot y_i \log(p_i) \; + \; \beta_n \cdot (1 - y_i)\log(1 - p_i)\right)$$, where:

N represents the number of outputs,

y_i represents the correct output,

p_i represents the predicted output,

β_p represents the weight of the positive part of the loss function,

β_n represents the weight of the negative part of the loss function

This function measures the average difference between the model's predictions and the correct predictions. A higher average difference indicates that the model is less accurate, which means that it needs to have stronger adjustments and vice versa. Classification models tend to favor predicting one class over the other (in this project's case, predicting possible binding too frequently). To address this bias, the loss function is multiplied by weights that reflect each class's contribution to the loss. This means that if the model is over-predicting the protein and candidate drug molecule will bind, there are more radical adjustments to the model parts that contributed to this decision.

The loss function is used to adjust the hidden layers' transformation mechanisms. Initially, the model makes predictions, which are then evaluated with a loss function. This resulting loss function is fed back to the hidden layers and adjust their parameters. This process repeats many times, often hundreds or thousands of times. There are some ways to

control the pace of training through some pre-set parameters before training. Specifically, the learning rate controls how significantly the model corrects its weights. A low learning rate requires more iterations to train a model but results in more precise adjustments. On the other hand, a higher learning rate may require fewer iterations, but the model is likely to miss optimal settings. For this model, the strategy was to keep a low learning rate and high number of iterations, to ensure accurate convergence.

The model was trained and tested on different datasets to ensure that the model is evaluated fairly. The *Database of Useful Decoys – Enhanced* (DUD-E) dataset (Mysinger, et al, 2012), which contains information on 102 target proteins along with candidate drugs (some bind, most don't), is used. The 102 target proteins were randomly split, with 15% used for testing and the remaining 85% for training. Each target protein and its corresponding candidate drug graphs were extracted and organized into datasets, with each protein target and candidate drug graph pair forming a row.

The performance of the model on training and testing datasets is recorded, and models that perform well on testing data are saved. The training results are analyzed to determine the best boundary between positive and negative outputs, and 0.13 was the best boundary. This means that if the model outputs a number greater than 0.13, it will be marked as a potential drug. The model's results are further analyzed and compared with other existing models using various performance metrics.

Results:

The model performance on the test dataset is evaluated on different evaluation metrics, as shown below:

Evaluation Functions	Results
ROC AUC Score	0.82
Enrichment Factor (top 1%)	22.36
Enrichment Factor (top 2%)	15.97
Enrichment Factor (top 5%)	9.67
Enrichment Factor (top 10%)	6.43
Enrichment Factor (top 20%)	4.08
Enrichment Factor (maximum possible)	35.99

Mean Squared Error (MSE)	0.18

ROC AUC Score is a measure of how well a model can differentiate between different categories, with a number between 0 and 1, with higher scores being better and 0.5 representing a model's performance through random guessing. Mean Squared Error (MSE) measures how much the predictions deviate from the correct answers, with lower scores being better (scikit-learn, *3.4. Metrics and scoring: quantifying the quality of predictions*). Enrichment Factor (EF) at top x% is a measure of how well a model's top x% predictions can identify candidate drug molecules within a large dataset, a metric especially relevant to the virtual screening problem (Wojcikowski, 2015). There's often also a maximum achievable Enrichment Factor that differs by dataset, as it changes according to the number of candidate drug molecules in a dataset and the size of the dataset.

To understand how successful the model is, the evaluation metrics are compared with those of other models in the literature, as shown below:

	ROC AUC	EF at 1%	EF at 2%	EF at 5%	EF at 10%	EF at 20%	EF max
TwinGNN	0.82	22.36	15.97	9.67	6.43	4.08	35.99
DLIGAND2	0.77	6.67	N/A	3.31	2.55	N/A	N/A
NF	0.90	N/A	N/A	N/A	N/A	N/A	N/A
ParaVS-ND	0.98	N/A	39.7	N/A	N/A	4.9	62.6

The TwinGNN row refers to this project's model performance

At a first glance, TwinGNN outperforms DLIGAND2 in terms of ROC AUC score, which indicates that the TwinGNN better differentiates between positives and negatives (binding potential). DLIGAND2 approximates binding energy, which simplifies the structure. A possible reason for why the TwinGNN model outperforms DLIGAND2 could be because DLIGAND2 may miss out on other characteristics that may be relevant to the problem. TwinGNN also outperforms DLIGAND2 on Enrichment Factor, but this is difficult to compare, as there is no maximum Enrichment Factor given in the paper.

Compared to NF and ParaVS-ND, TwinGNN doesn't perform as well in differentiating between positives and negatives, as seen by the ROC AUC score gap. The reason why NF and ParaVS-ND are successful may be because it specifically passes on the protein pocket's structure, where the drug molecules will likely bind, instead of passing the whole

protein. The whole protein may contain information that may not be as relevant to the protein-drug molecule binding problem in general. In terms of Enrichment Factor comparing TwinGNN to ParaVS-ND, a ratio of EF to EF max was taken for fair evaluation, shown below:

Model	EF to EF max at 2%	EF to EF max at 20%
TwinGNN	0.44	0.11
ParaVS-ND	0.63	0.08

ParaVS-ND outperforms TwinGNN at 2%, while TwinGNN outperforms ParaVS-ND at 20%. This indicates that the top 20% of TwinGNN's predictions are more accurate, whereas the top 2% ParaVS-ND's predictions are more accurate. Further investigation is needed to understand the differences between the results.

Overall, ParaVS-ND and NF outperform TwinGNN. They are different from TwinGNN in that they extract specific structural features, the protein pockets where the candidate drug molecule would likely bind, instead of passing on the whole protein structure, which may contain some irrelevant features. Despite this, Graph Neural Networks have a strong potential in solving virtual screening problems, as shown by the performance of ParaVS-ND, NF, and TwinGNN compared to the knowledge-based DLIGAND2. However, the advantage of DLIGAND2 is that one can understand why the algorithm gave its result, whereas it is hard to understand why TwinGNN and other machine learning algorithms gave their results.

Conclusion:

TwinGNN is a model that predicts whether a candidate drug molecule could bind with a target protein as part of virtual screening. It processes graphs of candidate drug molecules and target proteins through separate Graph Neural Networks, then passes the resulting graph embeddings to a Neural Network which predicts whether the candidate drug molecule can bind with the target protein. When compared with other models, TwinGNN outperforms a knowledge-based model known as DLIGAND2, by better differentiating between positives and negatives and showing a richer concentration of correctly predicted actives amongst its confident predictions. However, TwinGNN falls short against other Graph Neural Network-based models. ParaVS-ND and Neural Fingerprinting both extract

protein pockets explicitly, then perform similar operations. This could indicate that focusing on the protein pockets would enhance accuracy in predicting binding.

It is difficult, as well, to compare between TwinGNN and the other models in the paper solely on the DUD-E dataset. Some other datasets that TwinGNN could be evaluaed on are DUD-E+, CASF-2013, and PDBBind. An idea to explore in the future could also be to integrate knowledge-based methods alongside Graph Neural Networks to observe how known chemical knowledge could be enhanced with machine learned knowledge.

Acknowledgements:

I am deeply grateful to Lord Laidlaw and the team at the Laidlaw Foundation for the research and leadership development opportunities through the Laidlaw Scholars program. I also thank Dr. John Mitchell, my research supervisor, for his valuable time and advice. Finally, I would like to thank my family for their support and flexibility, which enabled me to pursue my Laidlaw plans.

References:

Cancer Research UK (2022) *How long a new drug takes to go through clinical trials*. Available at: https://www.cancerresearchuk.org/about-cancer/find-a-clinical-trial/how-clinical-trials-are-planned-and-organised/how-long-it-takes-for-a-new-drug-to-go-through-clinical-trials,

Chen, P. *et al.* (2019) 'DLIGAND2: An improved knowledge-based energy function for protein–ligand interactions using the distance-scaled, finite, ideal-gas reference state', *Journal of Cheminformatics*, 11(1). doi:10.1186/s13321-019-0373-4,

Gonczarek, A. *et al.* (2018) 'Interaction prediction in structure-based virtual screening using Deep Learning', *Computers in Biology and Medicine*, 100, pp. 253–258. doi:10.1016/j.compbiomed.2017.09.007,

DUD 'Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking', *Journal of Medicinal Chemistry*, 55(14), pp. 6582–6594. doi:10.1021/jm300687e,

Paszke, A. *et al.* (2019) *Pytorch: An imperative style, high-performance deep learning library*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Available at: https://doi.org/10.48550/arXiv.1912.01703.

Scannell, J.W. *et al.* (2012) 'Diagnosing the decline in pharmaceutical R&D efficiency', *Nature Reviews Drug Discovery*, 11(3), pp. 191–200. doi:10.1038/nrd3681,

scikit-learn (no date) *3.4. Metrics and scoring: quantifying the quality of predictions*. Available at: https://scikit-learn.org/stable/modules/model_evaluation.html,

Wouters, O.J., McKee, M. and Luyten, J. (2020) 'Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018', *JAMA*, 323(9), pp. 844–853. doi:10.1001/jama.2020.1166,

Wojcikowski, M. (2015) *Source code for oddt.metrics*, *oddt.metrics - Open Drug Discovery Toolkit 0.8 documentation*. Available at: https://oddt.readthedocs.io/en/latest/_modules/oddt/metrics.html,

Wu, J., Leng, D. and Pan, L. (2021) *ParaVS: A Simple, Fast, Efficient and Flexible Graph Neural Network Framework for Structure-Based Virtual Screening*, *arXiv.org*. Available at: https://arxiv.org/abs/2102.06086.