
Drug Toxicity Prediction Using Graph Neural Networks

Saranya Pal
Department of Chemistry
210930
saranyap21@iitk.ac.in

Monday 18th November, 2024

Abstract

Drugs play an extremely crucial role in saving lives of people. However, some of them are too toxic to be safe for human use. Determining a drug's toxicity usually takes a lot of time, as it involves testing on large groups of people, waiting to see side effects, and analyzing results. Apart from that, several chemical tests are conducted for testing a variety of results before arriving at a conclusion. This process can be extremely time-consuming delaying progress in the process of drug discovery. In this study, we aim to address this problem by using computational methods instead of traditional lab tests. Machine learning techniques are well-known for performing several classification tasks. We employ Graph Neural Networks(GNNs) to solve our problem. One thing must be noted is that for a classification to be performed by any ML model, a set of features must be provided from which the model will extract information and then perform the classification. However, in our case, we are provided with only the SMILES representation of the drug and the toxicity label of it. So, we start with extracting the feature vector from this SMILES representation (a type of chemical notation) of the drug. And then perform the classification using GNNs and attention mechanism to determine whether the drug is toxic or not. This study combines chemistry and machine learning, marking it a significant step forward in the process of drug discovery.

1 Challenges Faced

We have used the Clintox dataset for the entire analysis that we have performed here. The ClinTox dataset was obtained from `torch_geometric.datasets`. This dataset is part of the MoleculeNet dataset which we will discuss later. Working with the ClinTox dataset for predicting drug toxicity comes with several challenges:

- **Limited Features provided as Atomic Properties:** The dataset has only 9 atomic features per molecule, which often results in low accuracy, as the model is not able to capture any patterns from these 9 features. Also, in these type of problems, the structure of the molecule plays a really important role. And unless we encode the structure, we will not be able to improve the performance.
- **Class Imbalance:** There is also a big class Imbalance, with only 112 toxic molecules out of 1480 drugs, making it hard to correctly identify the toxic ones. It is obvious to us because among a sample of the dataset, only a small fraction will be toxic. And that is what we observe in the dataset.

As the provided features were insufficient, so we needed to extract features from its structure. Converting SMILES codes into useful features requires a lot of processing. Combining different embeddings like the x features, Node2Vec, and Graph2Vec, and making them work together in Graph Neural Networks (GNNs), adds more complexity.

2 Introduction and Problem Statement

The primary objective is to build robust ML model that could correctly predict whether a given drug is toxic or not from just the SMILES code of the drug as an input. Although this looks as a single problem, yet this involves two parts:-

- As discussed previously, the 9 atomic features provided are insufficient for an ML algorithm to learn patterns from. So, the graph structure must be encoded into meaningful features. To achieve this, we use the Node2Vec and Graph2Vec embedding methods.
 - Node2Vec is a node-level embedding technique that learns low-dimensional representations for nodes by sampling their local neighborhoods through biased random walks. It captures both local and global graph structures, balancing breadth-first (BFS) and depth-first (DFS) search strategies.
 - Graph2Vec generates a fixed-size vector representation for **entire graphs** by aggregating substructure information from their nodes and edges. It leverages a document embedding-like approach, treating each graph as a "document" and substructures as "words".
- We concatenate the embeddings from Node2Vec, Graph2Vec (after repeating it along axis=0) and x features to obtain the final embeddings. We believe that this embedding will provide a rich representation of our graph. Both the Node2Vec and Graph2Vec embeddings are used because the limitations of one embedding is fulfilled by the other. Thus, both of them are essential to obtain a rich representation of the graph.
- Now, we have a 64 dimensional feature vector for every atom of a molecule. From this feature matrix of molecules, we shall employ a Machine Learning model, especially Graph Neural Network (GNN) architectures and try to classify the drug as toxic or non-toxic.

3 Data Sources And Description

The dataset contains the drugs represented as SMILES code and their toxicity representation (0 represents non-toxic and 1 represents toxic). SMILES (Simplified Molecular Input Line Entry System) is a text-based notation to represent chemical structures using linear strings. There are 1480 drugs, out of which 112 of them are toxic. Each of the molecule can be represented as a Graph, with nodes as atoms and edges as bonds. In our representation, all the nodes are of same type and also all bonds are considered same and thus represented by the same type of bidirectional edge representation. The representation of a molecule along with its SMILES code is shown below:-

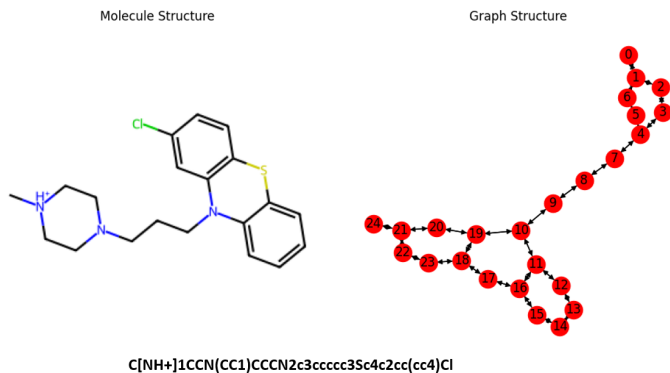


Figure 1: A Random Molecule

The ClinTox dataset is part of the MoleculeNet data, a collection of benchmark datasets designed for developing and evaluating machine learning models in molecular science, focusing on tasks like property prediction, toxicity assessment, and drug discovery. Apart from the ClinTox, several other datasets are part of it such as HIV, BACE, BBBP, Tox21, SIDER, MUV, PCBA, and ChEMBL, covering diverse tasks like solubility prediction, quantum mechanics, bioactivity classification, and toxicity analysis.

3.1 Train-Test Split

We used a train-test split of **80:20** for the training and then evaluating the performance of our models. After we have split our data, it is important to note the percentage of toxic drugs that were a part of the train data and the test data.


- **Train-Dataset:** After splitting the data into the above-mentioned ratio, the train dataset consisted of 1184 drugs, out of which 84 were toxic.
- **Test-Dataset:** The test dataset consisted of a total of 296 drugs out of which, 28 of the molecules were toxic.

Keeping a check on the no. of toxic drugs present in the train and test data separately is important for our problem. Due to the imbalance present in our dataset, there is a high chance that the model will be predicting that all drugs are non-toxic, which is undesirable. We will discuss more about it in the model evaluation.

4 Model Description

4.1 Graph Embedding

We have previously discussed that we have concatenated the x features, Node2Vec and Graph2Vec in the following manner as shown:-



0.17	-0.02	0.11	0.88	0.41	0.35	0.81	1
-0.11	-0.14	-0.01	-1.21	-0.41	0.77	0.22	2
-1.81	-0.97	0.25	-0.91	0.74	-0.11	-0.38	3
-1.25	0.33	0.05	0.18	0.22	0.35	-1.06	4
-1.57	0.04	-1.05	-1.07	0.50	-1.03	0.57	5

Figure 2: Visualisation of Graph Embedding

- The first 9 columns/features represent the x-features that was provided to us within the dataset. These features represent important properties of the atoms such as Atomic No., Chirality, Formal Charge, Hybridisation, Van Der Waal's Radius, No. of H atom, Implicit Valence, Degree and Is_Aromatic.
- The next 64 features that follow it are the embedding representations obtained from the Node2Vec algorithm.

- The final 64 features are the embeddings obtained from the Graph2Vec algorithm. Note that Graph2Vec converts the entire graph into a single feature vector. To match up the dimension, we concatenate the same representation along axis = 0 so that the dimensions are same now for all the embeddings.

Finally, we achieve a (n,137) dimensional matrix for each molecule/drug representing the graph. Now, instead of just using it directly in our model, we split the embedding into 7 different parts such that we could evaluate each individual embeddings separately and get a deeper intuition into the performance of each of these. The 9 sub-splits of the embeddings are done as:-

- x-features only (9-dim)
- Node2Vec only (64-dim)
- Graph2Vec only (64-dim)
- (x-features + Node2Vec) (73-dim)
- (x-features + Graph2Vec) (73-dim)
- (Node2Vec + Graph2Vec) (128-dim)
- (x-features + Node2Vec + Graph2Vec) (137-dim)

4.2 GNN Model Architectures

We now have 7 different embedding vectors. Now, to fit them into GNN models, we used a total of 6 different GNN architectures, each of them having its own advantages and limitations. The GNN architectures used in our study are:-

- **Simple GCN:** Utilizes convolution-like operations on the graph structure to learn node representations, capturing local interactions between atoms.
- **Simple GCN with Imbalance Handling:** Adjusts class imbalance during training by incorporating class weights into the loss function.
- **GAT with Imbalance Handling:** Uses attention mechanisms to prioritize important connections in the graph, focusing on the most informative interactions.
- **Transformers with Imbalance Handling:** Employs self-attention to capture long-range dependencies, improving performance for complex molecular structures.
- **SAGEConv with Imbalance Handling:** Aggregates neighbor information to generate scalable representations, suitable for large datasets.
- **ResGatedConv with Imbalance Handling:** Combines residual connections with gated units to address vanishing gradients and improve learning representations.

4.3 Inference

For each of these architectures, we used all of the 7 different embedding methods. Thus, we have built 7 different models per GNN architecture, resulting in a total of 42 different models! We compare the performance of all of these models in the Model Evaluation section.

4.4 Detailed Architecture

The architecture for each of the 42 models is as follows:- The first 3 layers of our model are GNN layers, with the following input and output dimension:-

- GNN1(input_dim, 16)
- GNN2(16, 8)
- GNN3(8, 8)

So, we obtain a matrix of (n, 8) dimensions. Now, we must use Pooling over the output from the GNN layer, before we put the output into the linear layers to perform the classification. We use

both, GlobalMaxPooling and GlobalAveragePooling over the obtained embeddings, and then concatenate the two. This results in a single 16-dimensional vector. [Dimension = (1,16)].

This output is now passed through three linear layers with the following details:-

- Linear1(16, 8)
- Linear2(8, 8)
- Linear3(8, 1)

The final Linear layer is also the output layer of our model. Finally, if the output probability is more than 0.5, we predict the drug as toxic (label=1), otherwise non-toxic.

4.5 Loss Function and Hidden Activations

The task with us is a Binary Classification task, where both the actual and the predicted labels are binary labels. For this reason, we use the binary cross-entropy loss function. We use `torch.nn.BCEWithLogitsLoss()` from PyTorch to implement this.

$$\text{Loss} = -[y \cdot \log(\sigma(x)) + (1 - y) \cdot \log(1 - \sigma(x))]$$

\tanh was used as the hidden activation for the GNN layers. For the linear layers, ReLu was used as the hidden activation except for the final layer. And as it is a binary classification task, we use Sigmoid activation for the final output layer, that predicts the probability of the drug being toxic.

5 Model Training

5.1 Class Imbalance Handling

As we discussed previously, the ClinTox dataset is highly imbalanced with just 10% of the drugs being labelled as toxic. So, to prevent our model from classifying all drugs as non-toxic, we train our models using Class-weights. Specifically, among those 6 architectures, 5 of them was trained using class weights. That means among the 42 models, 35 models were trained using Class Weights. This resulted in an improved prediction performance of our models.

5.2 Training Procedure

The models were trained with different no. of epochs in an attempt to prevent overfitting and achieving the best possible results. The no. of epochs ranged from 10 to 50 depending on the model.

6 Model Evaluation

6.1 Metrics Used

Since this is a classification task, metrics such as accuracy, precision, recall, F1score and ROC-AUC scores were used. In most of the classification tasks, accuracy is considered a reliable metric. However, this is not true in our case due to the imbalance that the data poses. The best metric for our purpose is ROCAUC score and models are judged on the basis of this.

The reason behind this is that, even if we declared that all drugs are nontoxic, we will achieve an accuracy of more than 90%, which is misleading since the model fails to classify any drug as toxic.

However, in the same situation, if we used ROCAUC score, classifying all drugs as nontoxic will never yield a score higher than 0.50. To achieve a score higher than it, the model must correctly classify some toxic drugs as well..

6.2 Results

6.2.1 Simple GCN

The results for the Simple GCN layer is as follows:

Table 1: GCN Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.9054	0	0	0	0.5
Using Node2Vec only	0.9054	0	0	0	0.5
Using Graph2Vec only	0.9054	0	0	0	0.5
Using (x + Node2Vec)	0.9054	0	0	0	0.5
Using (x + Graph2Vec)	0.9054	0	0	0	0.5
Using (Node2Vec + Graph2Vec)	0.9054	0	0	0	0.5
Using (x + Node2Vec + Graph2Vec)	0.9054	0	0	0	0.5

We can easily interpret the results here that all the 7 models fails to classify any drug as toxic. All drugs are classified nontoxic. Thus, the results from this model makes no sense.

6.2.2 GCN with Class Weights

The results for the GCN with Class Weights model is as follows:

Table 2: GCN with Imbalance Handling Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.7939	0.1333	0.2143	0.1644	0.5344
Using Node2Vec only	0.5541	0.1	0.4643	0.1646	0.5139
Using Graph2Vec only	0.9122	1	0.0714	0.1333	0.5357
Using (x + Node2Vec)	0.7095	0.1705	0.5357	0.2586	0.6317
Using (x + Graph2Vec)	0.9122	1	0.0714	0.1333	0.5357
Using (Node2Vec + Graph2Vec)	0.7264	0.1268	0.3214	0.1818	0.545
Using (x + Node2Vec + Graph2Vec)	0.4797	0.1294	0.7857	0.2222	0.6167

Interpreting the results above, we can clearly see that in this model, using (x + Node2Vec) yields the best possible results with the highest ROCAUC score. This ROCAUC score is also the highest among all the models, thus the best model in our study.

6.2.3 GAT with Class Weights

The results for the GAT with Class Weights model is as follows:

Table 3: GAT with Imbalance Handling Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.8818	0.1818	0.0714	0.1026	0.5189
Using Node2Vec only	0.6115	0.115	0.4643	0.1844	0.5456
Using Graph2Vec only	0.9054	0	0	0	0.5
Using (x + Node2Vec)	0.7162	0.1216	0.3214	0.1765	0.5394
Using (x + Graph2Vec)	0.8953	0.3636	0.1429	0.2051	0.5584
Using (Node2Vec + Graph2Vec)	0.6115	0.1217	0.5	0.1958	0.5616
Using (x + Node2Vec + Graph2Vec)	0.7331	0.1077	0.25	0.1505	0.5168

Despite the Attention mechanism used, this model does not show an improvement in performance and performs average when we look at the ROCAUC scores and the accuracy.

6.2.4 Transformers with Class Weights

The results for the Transformers with Class Weights model is as follows:

Table 4: Transformers with Imbalance Handling Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.9122	1	0.0714	0.1333	0.5357
Using Node2Vec only	0.9054	0	0	0	0.5
Using Graph2Vec only	0.9054	0	0	0	0.5
Using (x + Node2Vec)	0.7822	0.1698	0.3214	0.2222	0.5786
Using (x + Graph2Vec)	0.9054	0.5	0.0714	0.125	0.532
Using (Node2Vec + Graph2Vec)	0.6723	0.0633	0.1786	0.0935	0.4512
Using (x + Node2Vec + Graph2Vec)	0.5777	0.124	0.5714	0.2038	0.5749

Using Transformers architecture performed better on average than the GAT model used previously. However, this failed to become the best possible model despite the complex architecture. And stays as the second best model, which is achieved when (x + Node2Vec) is used as the embedding.

6.2.5 SAGEConv with Class Weights

The results for the SAGEConv with Class Weights model is as follows:

Table 5: SAGEConv with Imbalance Handling Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.0946	0.0946	1	0.1728	0.5
Using Node2Vec only	0.7162	0.1316	0.3571	0.1923	0.5554
Using Graph2Vec only	0.9122	1	0.0714	0.1333	0.5357
Using (x + Node2Vec)	0.75	0.129	0.2857	0.1778	0.5421
Using (x + Graph2Vec)	0.9122	1	0.0714	0.1333	0.5357
Using (Node2Vec + Graph2Vec)	0.6486	0.1122	0.3929	0.1746	0.5341
Using (x + Node2Vec + Graph2Vec)	0.7196	0.1333	0.3571	0.1942	0.5573

Again, the models trained here performed average on the basis of it's ROCAUC score.

6.2.6 ResGatedConv with Class Weights

The results for the ResGatedConv with Class Weights model is as follows:

Table 6: ResGatedConv with Imbalance Handling Results

Embedding Type	Accuracy	Precision	Recall	F1-score	ROC
Using only x	0.9054	0	0	0	0.5
Using Node2Vec only	0.9054	0	0	0	0.5
Using Graph2Vec only	0.9054	0	0	0	0.5
Using (x + Node2Vec)	0.6858	0.1264	0.3929	0.1913	0.5546
Using (x + Graph2Vec)	0.9054	0	0	0	0.5
Using (Node2Vec + Graph2Vec)	0.7939	0.1333	0.2143	0.1644	0.5344
Using (x + Node2Vec + Graph2Vec)	0.7703	0.1429	0.2857	0.1905	0.5533

Similar to the last model, this model also fails to achieve top performance among these models and performs average.

6.3 Visualization of the Results

Ideally, the results should be visualized as bar plots. However, this may result in a total of 30 plots in our case, which might really be overwhelming. Due to this reason, we used lineplots to plot the results of all different models in just 5 plots, each representing a unique metric. The plots are as follows:-

6.3.1 Accuracy Plot:

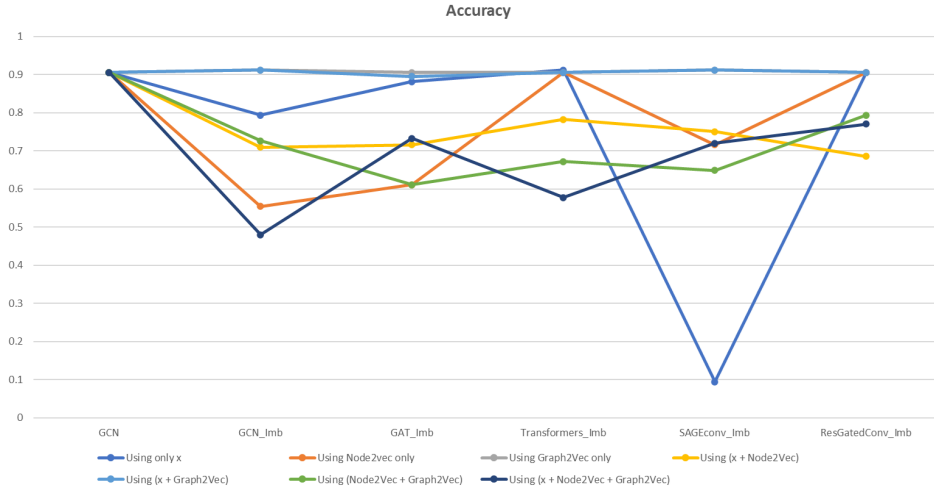


Figure 3: Accuracy Comparison Plot

6.3.2 Precision Plot:

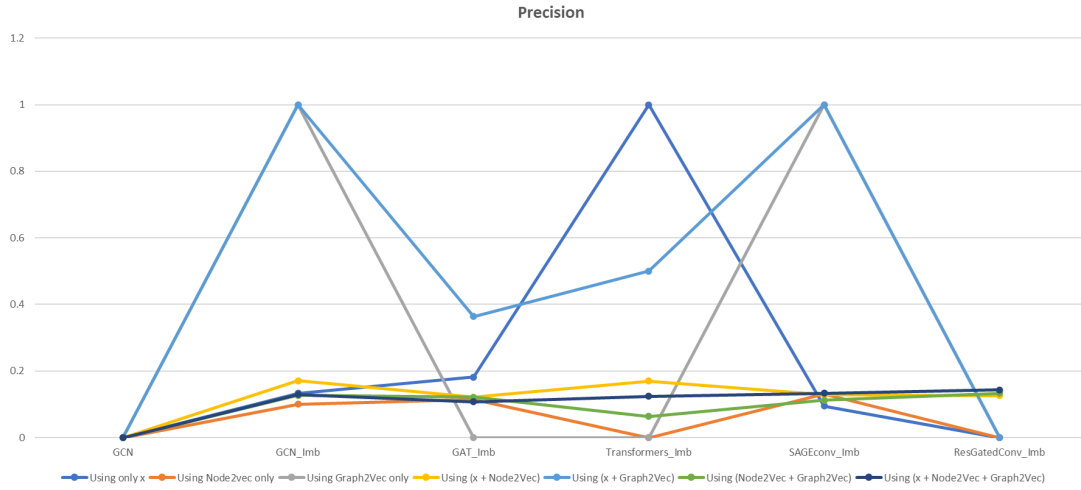


Figure 4: Precision Comparison Plot

6.3.3 Recall Plot:

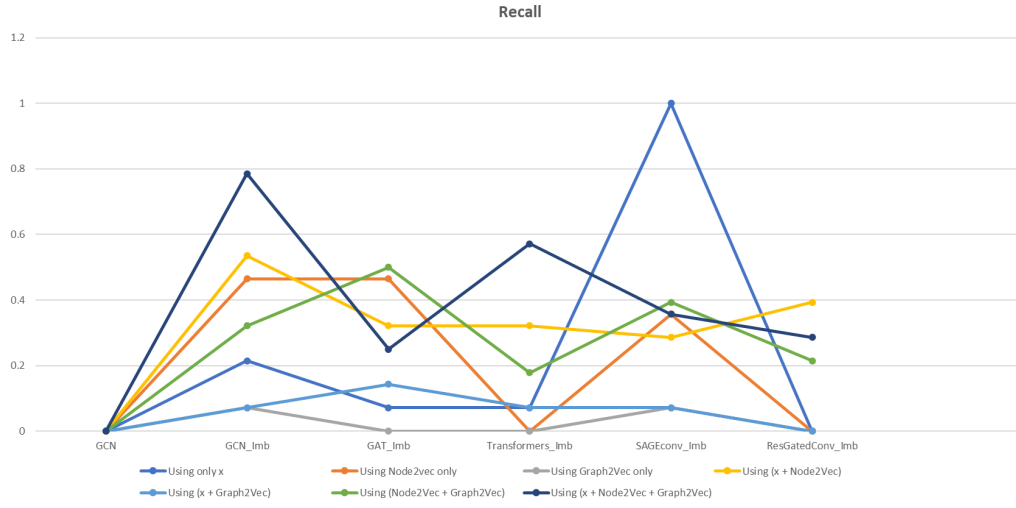


Figure 5: Recall Comparison Plot

6.3.4 F1-score Plot:

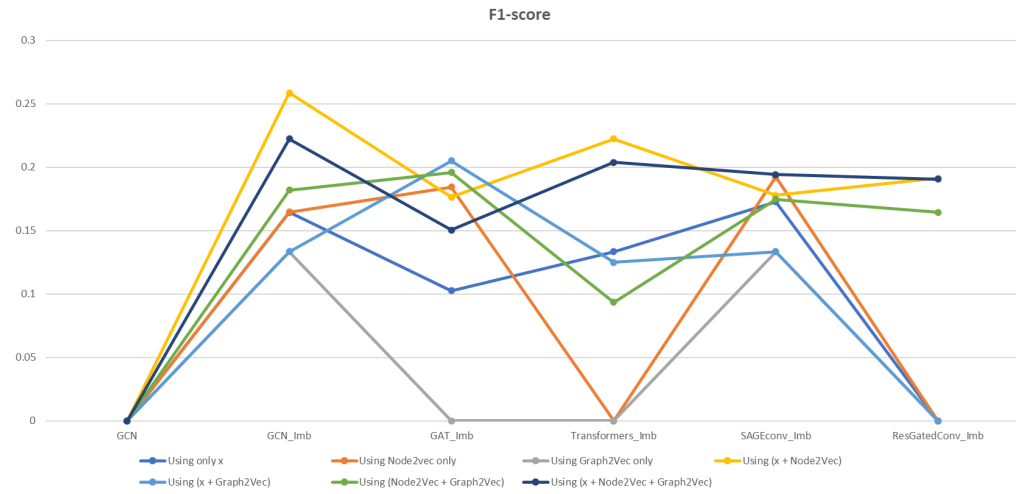


Figure 6: F1-score Comparison Plot

6.3.5 ROC-AUC score Plot:

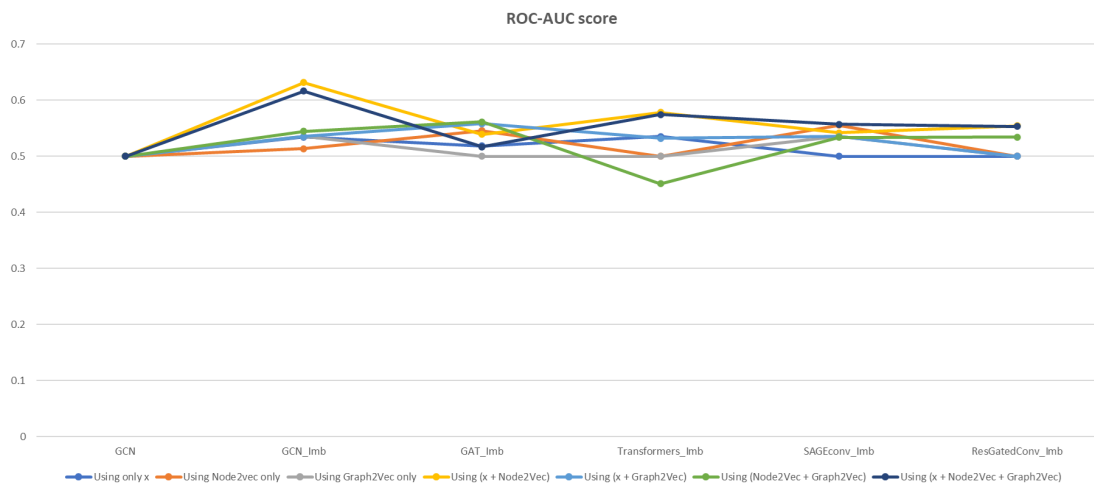


Figure 7: ROC-AUC score Comparison Plot

7 Conclusion

Graph Neural Network (GNN) architectures have demonstrated remarkable effectiveness in leveraging graph structures for toxicity classification tasks, making them a powerful tool in drug discovery. It is also show that encoding the structure performs better than by not encoding them and using other methods to extract features. By integrating unique embedding strategies, such as combining atom-level features x with Node2Vec and Graph2Vec embeddings, the model achieved significant performance gains, enhancing its predictive accuracy. This approach accelerates drug discovery by offering a computationally efficient alternative to traditional laboratory tests, reducing costs and time while maintaining high reliability in toxicity prediction.

8 Future Work

Although significant analysis has been done on how Machine Learning, especially Graph Neural Networks could be employed for Drug Toxicity Prediction, still there are scopes for significant improvements. Here are some of the few works that could be experimented with in future:

1. **Node Types:** Currently, all nodes in the graph are of the same type, failing to distinguish between different atom types. To address this, a heterogeneous graph representation is needed instead of the current homogeneous graph.
2. **Hydrogen Atoms:** Hydrogen atoms are not included in the graph structure, which may introduce errors in predictions. Their inclusion could improve the accuracy of the model.
3. **Class Imbalance:** The dataset suffers from class imbalance, and conventional methods like SMOTE are unsuitable for this domain. Problem-specific strategies need to be developed to handle this issue effectively.
4. **Heterogeneous Architectures:** Exploring heterogeneous graph architectures, such as SimpleRelationalConv, could enhance the model. Alternatively, novel, problem-specific architectures might yield better results.
5. **Bond Representation:** All bonds are currently represented as bidirectional edges, which do not capture the nuances of double and triple bonds. More accurate bond representations are required.
6. **Molecule-Specific Properties:** Properties unique to molecules, such as chiral sub-graphs, 3D structure of the molecule etc, play a significant role in drug toxicity. These features need to be encoded to improve the model's predictive capabilities.