# Predicting Toxicity of a Molecule Using Graph Neural Networks: Midterm Report

Isaiah Gocool
York University
Toronto, Canada
218918052
goisaiah@my.yorku.ca

Vince Flores
York University
Toronto, Canada
218801779
vincegab@my.yorku.ca

**ACM Reference Format:**
Isaiah Gocool and Vince Flores. 2025. Predicting Toxicity of a Molecule Using Graph Neural Networks: Midterm Report. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

### 1.1 Motivation

The research and development of new medicinal drugs is a very long and resource-intensive process, most notably being burdened by the toxicity of molecules [1]. A survey conducted by the Congressional Budget Office in 2021 found that the expected cost of developing a new drug ranges from $1 billion to more than $2 billion, and only 12% of the drugs entering clinical trials are approved by the FDA [2]. These statistics highlight the need for stronger predictive models that can determine a drug's toxicity early in the research and development process. Many experiments have been conducted on the toxicity of specific molecules, linking this property to molecular structures. The structure of a molecule can be easily modeled through the use of graphs, where the nodes represent the atoms and the edges represent the bonds. Predictive models can be applied to graphs to better understand trends, analyze complex networks, and find new connections between features. Neural networks are one such example of a predictive model that can be applied to graphs. In this project, our aim is to use two graph neural networks (GNNs), specifically graph convolution networks (GCNs) and graph attention networks (GATs), to predict if a molecule is toxic on the basis of its molecular structure.

### 1.2 Why Graph Neural Networks Were Chosen

Graph Neural Networks were chosen as an approach to solve this problem because the structure of molecules can be naturally represented by graphs, as mentioned earlier. GNNs learn features directly from the molecule's structure, capturing how chemical environments and functional groups impact toxicity. GCNs can act as a baseline for how the GNNs should behave, capturing basic structures like bonding patterns. GATs are able to emphasize chemically important neighbouring atoms, showing which interactions have the greater impact on toxicity and making it a more chemically realistic model. By comparing the results from the two models, we can determine whether a molecule's basic molecular structure is responsible for toxicity, or if specific context-dependent atoms are the cause.

### 1.3 Related Work

One of the traditional methods for predicting molecular compounds is through relying on "molecular fingerprints", such as Extended-Connectivity Fingerprints (ECFP). Molecular fingerprints are fixed-length vectors that represent a molecule's structure. These vectors can then be used as inputs for a machine learning algorithm like Random Forests for analysis [3]. A major downside of this method is that the conversion of a three-dimensional structure into a two-dimensional one inherently leads to a loss of information, something graphs can prevent from occurring.

Zhang et. al (2025), provides an enhancement in molecular property predictions by introducing MTSSMol, a multitask self supervised deep learning framework. The tool involves a GNN encoder which takes in molecules represented in Simplified Molecular Input Line Entry System (SMILES) notation [4] as input and outputs latent feature representations or embeddings which are then useful for performing predictive tasks. The nodes of the GNN represent atoms while the edges represent bonds. The advantage of using a GNN is that it allows aggregation and integration operations to extract features from nodes and its neighbors. The study found improvements over classical fingerprint methods like ECFP [5].

There are two types of graph neural networks that are relevant to our project. Graph Convolution Networks (GCN) as proposed by Kipf & Welling (2017) focus on aggregating information from a node's neighbours, helping it to detect simple patterns [6]. For each node in the graph, a GCN creates a new feature vector by averaging the feature vectors of itself and its neighbours. This will not only give us information on an atom but also its immediate chemical environment in the molecule, such as functional groups. The second GNN of interest is Graph Attention Networks (GAT) by Veličković et al (2018). GATs are a more complex version of GCNs that assign a weight to each node, so information obtained from a node's neighbours also has a level of importance assigned to it [7]. This can help us detect which atoms have a greater influence on factors like toxicity.

### 1.4 Problem Definition

The core problem we are trying to solve is to learn the relationship between a molecule's structure (atoms and bonds) and its function


*Conference'17, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

through the use of graph neural networks. Some questions this project aims to answer include the following:

- Are GNNs better at predicting molecular properties than standard machine learning methods, like random forests?
- Which GNN is better for this task?
- What are the limitations of GNNs when predicting molecular properties?

Although the primary goal of this project is to predict the toxicity of a molecule, we hope to expand it to predict other molecular properties as well if time permits.

## 2 Methodology

The following steps will be followed to build, compare, and evaluate predictive graph models.

### 2.1 Data Acquisition and Pre-Processing

We will be using a dataset from the website MoleculeNet, introduced by Wu, et. al (2017), which curates large scale benchmarks for machine learning [8]. The Tox21 dataset is used for training, testing, and validating the GNNs, since it contains molecules that have been deemed toxic or non-toxic based on a set of 12 experiments [9]. A preview of Tox21 has been included in figure 1, showing what the first 5 rows of the dataset look like.

| | NR-AR | NR-AR-LBD | NR-AhR | NR-Aromatase | NR-ER | NR-ER-LBD | NR-PPAR-gamma | SR-ARE | SR-ATAD5 | SR-HSE | SR-MMP | SR-p53 | mol_id | smiles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 1.0 | NaN | NaN | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | TOX3021 | CCOc1ccc2nc(S[N](=O)=O)sc2c1 |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | NaN | 0.0 | NaN | 0.0 | 0.0 | TOX3020 | CCN1C(=O)NC(c2ccccc2)C1=O |
| 2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 0.0 | NaN | 0.0 | NaN | NaN | TOX3024 | CC[C@]1(O)CC(=O)O[C@@H]1CCC1=CCCC[C@@H](C)[C@H]... |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | NaN | 0.0 | NaN | 0.0 | 0.0 | TOX3027 | CCCN(CCC)C(=O)c1cc(S(=O)(=O)Nc1ccc(C)cc1C |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | TOX20800 | CC(O)(P(=O)(O)O)P(=O)(O)O |

**Figure 1: Preview of the Tox21 dataset.**

The first 12 columns (NR-AR to SR-p53) are the names of toxicity targets, which are proteins, receptors, or cellular pathways. The activation of toxicity targets could indicate possible toxicity. If any column is labeled '1', that means the molecule is toxic. This is followed by a column for the molecule ID, and lastly a column for SMILES. SMILES are ASCII representations of molecular structures, with symbols representing bonds and letters representing atoms. For example, the SMILES representation of acetic acid is `CC(=O)O`. The RDKit library is fully compatible with SMILES and can be used to directly create graphs from them, as described in section 2.3 of this report. An outline of how data preprocessing is handled can be seen in algorithm 1.

**Algorithm 1** Data preprocessing algorithm

```
df ← dataset
while rows in df do
    if SMILES is valid then
        Add SMILES to list of valid SMILES
        Add label to list of valid labels
    else
        Skip molecule
    end if
end while
Split data for training, validation, and testing
Save preprocessed data to .pkl files
```

In order to get the dataset ready for training the models, missing values and invalid molecules are handled through data cleaning and data reduction. After some pre-processing the dataset is reduced to 7823 molecules from the original 7831. The data is then divided into $\tilde{8}0\%$ training, $\tilde{1}0\%$ validation, and $\tilde{1}0\%$ testing sets. Further fine-tuning on these parameters will be performed for the final version of this project to see if more accurate results can be produced. The new datasets are then saved to their own PKL files so they could be used in separate Jupyter Notebooks.

Further data transformation is done depending on the model (e.g. converting a molecule to a graph) and will be discussed in later sections.

### 2.2 Random Forest Classification Model Implementation

To address the need for a machine learning model, a Random Forest Classifier (RFC) from Scikit-learn library is used, with a Gini index strategy and 100 trees in the forest. The RFC model is trained using a dataset of ECFP representation of a SMILES represented molecule. Each molecule is labeled either 1 for toxic and 0 for non-toxic. The label is determined to be toxic if at least one of experiments from the set of binary valued attributes/experiments conducted on the molecule is toxic. Equation 1 shows how the label is assigned where $label_i$ is the label for a molecule and $X_i$ is the set of experiments conducted on the molecule.

$$label_i = max(X_i) \tag{1}$$

$$x_i \in X_i = \begin{cases} 1 & \text{if } molecule \text{ is toxic} \\ 0 & \text{if } molecule \text{ is non-toxic} \end{cases} \tag{2}$$

### 2.3 GNN Model Implementation

Molecules are represented by SMILES strings, and they will be converted into graph data structures using the PyTorch Geometric library. Nodes will represent atoms, and edges will represent bonds. Node features include atom type, charge, and hybridization, while edges only have the bond type as their sole feature. The two GNNs (GCNs and GATs) will aggregate the data on the molecules to produce a feature summarizing whether a molecule is toxic.

```python
def build_graph(smi, labels):
    try:
        data = from_smiles(smi)
        data.y = torch.tensor(labels, dtype=torch.float)
        return data
    except Exception:
        return None


def make_dataset(smiles_list, label_matrix):
    dataset = []
    for smi, lbl in zip(smiles_list, label_matrix):
        g = build_graph(smi, lbl)
        if g is not None:
            dataset.append(g)

    return dataset
```

**Figure 2: Code used to create the graph objects**

Before implementing the GNNs, we had to create the graphs that would be used to model the molecules. To do this, we used the machine learning library PyTorch. The full code can be seen in figure 2. The function `build_graph` creates a single graph for a molecule, which is represented by the `data` variable. The `from_smiles` function from the RDKit library is called when initializing `data` because it is RDKit's built-in function to directly convert SMILES to a PyTorch graph object. This function also automatically makes the atoms nodes and bonds the edges. `data.y` uses the `torch.tensor` function to create a vector that stores all the toxicity labels for the graph. The `make_dataset` function is used to create a list that stores all the graphs.

Algorithm 2 shows the outline of how the GNN algorithms are implemented. The model can use either GCN or GAT layers, which changes how the information is processed. Regardless, both layers initially perform a function known as message passing, which updates each atom's feature by combining information from neighbouring atoms.

**Algorithm 2** GNN algorithm

**Ensure:** Predicted toxicity scores $\hat{y}$
  $h \leftarrow x$
  **for** each GNN layer **do**
    **if** *model_type* = GCN **then**
      $h \leftarrow \text{GCNConv}_\ell(h, \textit{edge\_index})$
    **else if** *model_type* = GAT **then**
      $h \leftarrow \text{GATConv}_\ell(h, \textit{edge\_index})$
    **end if**
    $h \leftarrow \text{ReLU}(h)$
    $h \leftarrow \text{Dropout}(h)$
  **end for**
  $h_{\text{mol}} \leftarrow \text{GlobalMeanPool}(h, \textit{batch})$
  $\hat{y} \leftarrow \text{MLP}(h_{\text{mol}})$
  **return** $\hat{y}$

In GCNs, all neighbouring atoms are treated as equally important, so no weighting is applied to them. Equation (3) shows how GCNs update an atom $v$. However, the algorithm cannot determine which atoms matter more chemically or contribute more to toxicity. GATs solve this problem by computing an attention weight for each bond/edge, and then using that weight to boost important neighbouring atoms, while suppressing less important ones. Equation (4) shows how GATs update an atom $v$, where $\alpha_{vu}$ represents the attention weight.

$$h'_v = \text{ReLU}\left( W \cdot \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} h_u \right) \tag{3}$$

$$h'_v = \text{ReLU}\left( \sum_{u \in \mathcal{N}(v)} \alpha_{vu} W h_u \right) \tag{4}$$

After calculating the scores, both algorithms apply the Rectified Linear Unit (ReLU) function after each message passes. Once message passing is performed, an atom/node's feature vector gets new values which can be positive, negative, or zero. The ReLU function is responsible for filtering through the values in that feature vector and changing all the negative values to zero. This does not entirely eliminate activation values in feature vectors, but it filters out internal representation components that are not deemed important to the model's toxicity prediction at that time.

The dropout function is only used during training, and it randomly turns off neurons in the GNN by zeroing out a fraction of values in an atom's feature vector. This prevents the model from overfitting, so it does not rely too strongly on single learned features. It also improves generalization, making it more adaptable for future predictions.

The Global Mean Pool (GMP) and Multi-Layer Perceptron (MLP) functions are used to make the final prediction for the entire molecule. GMP takes the average of all the atom feature vectors, and combines the results into a single vector that represents the molecule/-graph. MLP is a simple neural network that uses the vectors from GMP to make a final prediction on the 12 toxicity targets for each molecule.

# 3 Evaluation

## 3.1 Datasets

For testing, we used 20% of the Tox21 dataset, 10% test set and 10% validation set which are 783 and 782 molecules respectively. The pre-processing of converting SMILES to graph or ECFP are done on both test sets.

A problem worth noting for evaluation of RFC is that the dataset is imbalanced due to the number of non-toxic labeled molecules being larger in each of the dataset split. This may be due to poor handling of missing values, which should be investigated in the future.

## 3.2 Experiments

*3.2.1 Experiment 1.* The first experiment will be the implementation of the two GNNs, GCNs and GATs, and a comparison of their accuracy. This experiment involves training the model for 50 epochs, which means it does a pass through of all molecular graphs 50 times, adjusting weights each time to reduce precision error. This is done for both GCNs and GATs, resulting in 50 epochs for each. The validation loss is calculated during training in order to tune the model. The test loss is only calculated once, after training is done to report the final performance. After testing is finished on all epochs, the best results from each model are recorded to check their accuracy. The results can be seen in table 1.

| Model | Validation Loss | Test Loss |
|---|---|---|
| GCN | 0.2149 | 0.2323 |
| GAT | 0.2139 | 0.2305 |

**Table 1: Experiment 1 Results: Comparing GCN and GAT Performance**

*3.2.2 Experiment 2.* The second experiment will compare the performance of the baseline model to the GNNs. This will be done by running each model separately on the same data. The second

experiment will compare the performance of the two GNN models, the GCN and the GAT. An error analysis will be performed during all experiments, as well as standard machine learning metrics like accuracy, precision, recall, and F1-score. Table 2. shows the performance metrics of the GNN models in comparison with the baseline RFC.

| Model | Accuracy | ROC$_AUC$ | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| GCN | 0.7677 | - | - | - | - |
| GAT | 0.7695 | - | - | - | - |
| RFC | 0.7216 | 0.6871 | 0.7116 | 0.6871 | 0.6924 |

**Table 2: Experiment 2 Results: Comparing GNN models with RFC model**

The results show that the three models are close in accuracy, but with the GNN models being more accurate than the RFC model especially the GCN model.

## 4 Conclusions and Future Work

To recap the work we have done so far, we have created two experiments to test our model for predicting toxicity in molecules. The first model involves implementing the GCN and GAT models, and comparing the accuracy of the two models. The second experiment is similar, but it involves implementing the baseline model, a Random Forest Classifier with ECFP representation of molecules. The results from the baseline model are then compared to the results from the GNN models.

The GNN models have been fully implemented and the overall accuracy is impressive. The GCN model performed with an accuracy of 76.77%, and the GAT model with 76.95%. As expected, the GAT model was more accurate, but the results are very close. Further fine-tuning of the model parameters will be conducted in the future to see if accuracy can be improved. One major flaw with the current implementation of the GNNs is that all NaN labels are set to 0, which is an incorrect assumption to make since those toxicity targets may not necessarily be non-toxic, they just have not been tested yet. Future iterations of the model will need to take this into account by masking the loss. We also plan on using the RDKit to visualize which atoms or functional groups contribute the most to toxicity, and extrapolating how the GNNs made their predictions.

Future improvements may also involve performing k fold cross-validation on the dataset. The idea is that each experiment will have the opportunity to train and test on every part of the dataset. With this method and before training each model, the dataset will be split into $k$ folds , then iteratively use each fold as a training set while the other $k-1$ folds will be used as testing sets. A benefit will be analyzing a set of results instead of just one.

## References


[1] Duxin Sun, Wei Gao, Hongxiang Hu, and Simon Zhou. Why 90% of clinical drug development fails and how to improve it? *Acta Pharmaceutica Sinica B*, 12, 02 2022.

[2] David Austin and Tamara Hayford. Research and development in the pharmaceutical industry, 04 2021.

[3] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50:742–754, 04 2010.

[4] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28:31–36, 02 1988.

[5] Xin Yang, Yang Wang, Ye Lin, Mingxuan Zhang, Ou Liu, Jianwei Shuai, and Qi Zhao. A multi-task self-supervised strategy for predicting molecular properties and fgfr1 inhibitors. *Advanced Science*, 12, 02 2025.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv (Cornell University)*, 01 2016.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv (Cornell University)*, 10 2017.

[8] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: A benchmark for molecular machine learning, 10 2018.

[9] Zhenqin Wu, Bharath Ramsundar, Evan Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh Pappu, and Karl Leswing. Datasets.