

Predicting Toxicity of a Molecule Using Graph Neural Networks: Final 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: Final Report. In . ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Abstract

Toxicity of molecules is a common issue affecting drugs in the pharmaceutical industry. Significant work has been done on trying to accurately predict molecule toxicity before drugs are developed, where small improvements can result in major reductions to experimental costs. In this project, implemented two graph neural network (GNN) architectures, a graph convolutional network (GNN) and a graph attention network (GAT), to model molecular structures and predict their toxicity using the Tox21 dataset. Molecules are represented by SMILES strings in the dataset, and graph structures are built directly from them. RandomForest was implemented as the baseline model, and results from all three models were compared using different metrics such as precision, accuracy, and recall. Both GNN models were found to perform better than RandomForest, with GAT specifically having the highest accuracy. The GNN models achieve a similar performance but have different approaches, highlighting how specific relationships between atoms and bonds can contribute to toxicity.

2 Introduction

2.1 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.

2.2 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. GNNs are also useful for multi-label classification tasks, which is important for predicting the 12 toxicity targets found in the Tox21 dataset we used. Within the domain of this problem, GNNs have an advantage over traditional graphs in that traditional graph algorithms treat graphs as pure topology. That is not ideal for solving this problem, as the graphs lose all chemical meaning with those algorithms. With GNNs, they can learn patterns from data embedded in the graphs, retaining the chemical context of the atoms and molecules.

2.3 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

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.

2.4 Problem Definition

The core problem we are trying to solve is to learn the relationship between a molecule's structure (atoms and bonds) and how it influences toxicity through the use of graph neural networks. Some questions that this project aims to answer include the following:

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

3 Methodology

The following section explains how we built, compared, and evaluated predictive graph models.

3.1 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-LD	NR-AR	NR-Aromatase	NR-ER	NR-ER-LD	NR-PAR-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	Tox13011	CCOC1=CC=CC=C1C(=O)O
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	0.0	NaN	0.0	0.0	Tox13020	CCOC1=CC=CC=C1C(=O)O
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	0.0	NaN	NaN	Tox13024	CCOC1=CC=CC=C1C(=O)O
3	0.0	0.0	0.0	0.0	0.0	0.0	NaN	0.0	NaN	0.0	0.0	0.0	Tox13027	CCOC1=CC=CC=C1C(=O)O
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	Tox13030	CCOC1=CC=CC=C1C(=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 80% training, 10% validation, and 10% testing sets. 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.

3.2 Random Forest Classification Model Implementation

To address the need for a machine learning model, an ensemble model is chosen such as the Random Forest Classification model (RFC). It is trained by building x number of decision trees, where each tree is mutually exclusive with other trees and is responsible for casting a vote for a class [10]. Since the model is made of many decision trees, overfitting is avoided. The experiment uses the RandomForestClassifier from the Scikit-learn library with 1000 trees in the forest, and class-weight w_i set to "balanced" to account for imbalance of tox21 dataset [11]. The model is also configured to handle multiple labels helpful for experiment 2.

The RFC model is trained using ECFP representation of SMILES molecule as the features and the 12 labels as classes.

The concept behind the conversion of SMILES to ECFP is outlined by Algorithm 2.

Algorithm 2 SMILES to ECFP algorithm

```

function CONVERTToECFP(SMILES)
  if SMILES.isLengthOne then
    return SMILES.atom.info
  end if
  while radius not 3 do
    for atom in SMILES do
       $ecfp \leftarrow ecfp \vee atom.info \vee$ 
      CONVERTToECFP(SMILES.neighbors[radius])
    end for
    radius ++
  end while
  return ecfp
end function

```

The algorithm takes in a SMILES represented molecule, recursively takes information from every atom and its neighbors. Then encodes it into a binary string of length 2048. Each bit represents information such as the type of atom, position, or bond-type.

3.3 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. GNNs function by initially having atoms/nodes exchange information from their feature vectors with each other to create new GNN layers where atoms have information about atoms from 2-3 bonds away. Afterward, the global pooling layer aggregates the data from all of the atoms (by computing the mean) to create a single vector for the entire molecule. The MLP (Multi-Layer Perceptron) then uses that vector to create predictions for the 12 toxicity targets for the molecule/graph as a whole.

```

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. Figure 3 uses molecule 260 as an example of a Lewis structure diagram of a molecule and its graph representation.

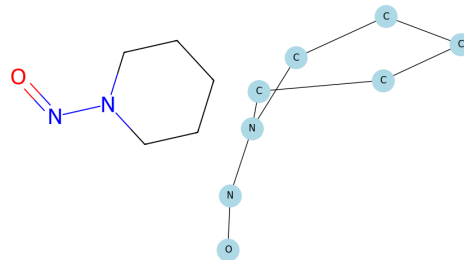


Figure 3: A drawing of molecule 260 along with its graph representation

Algorithm 3 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 3 GNN algorithm

```

Ensure: Predicted toxicity scores  $\hat{y}$ 
 $h \leftarrow x$ 
for each GNN layer do
  if  $model\_type = GCN$  then
     $h \leftarrow GCNConv_{\ell}(h, edge\_index)$ 
  else if  $model\_type = GAT$  then
     $h \leftarrow GATConv_{\ell}(h, edge\_index)$ 
  end if
   $h \leftarrow ReLU(h)$ 
   $h \leftarrow Dropout(h)$ 
end for
 $h_{mol} \leftarrow GlobalMeanPool(h, batch)$ 
 $\hat{y} \leftarrow MLP(h_{mol})$ 
return  $\hat{y}$ 

```

In GCNs, all neighbouring atoms are treated as equally important, so no weighting is applied to them. Equation (1) shows how GCNs update an atom v , where h'_v represents the updated node v , $N(v)$ represents the neighbours of v , and W represents the learnable weight matrix. As a drawback, the GCN 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 (2) shows how GATs update an atom v , where α_{vu} represents the attention weight.

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

$$h'_v = \text{ReLU} \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} W h_u \right) \quad (2)$$

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.

4 Evaluation

4.1 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.

4.2 Evaluation Metrics

4.2.1 AUC-ROC. Area Under the Curve (AUC) and Receiver Operating Curve (ROC) can be combined to result in an AUC-ROC curve, a metric used to evaluate the performance of binary classification models. Tox21 is a binary classification model since it assigns 1 or 0 to each toxicity target, representing if the molecule activates that target. This metric is primarily used for the dataset because Tox21 contains an imbalance of toxic and non-toxic molecules, so it is essential that our model performs well at separating positives from negatives. An ROC curve plots the relationship between the true positive rate and false positive rate. AUC-ROC condenses the results from the plot into a single value between 0 and 1 by computing the area under the ROC curve. Values closer to 1 mean the model is an accurate classifier.

4.2.2 Accuracy. The accuracy is used to measure how many molecules are correctly classified as toxic from the set of input molecules.

4.2.3 Precision. Precision is used to measure how often the models can correctly classify a molecule as toxic from the set of molecules that are actually toxic.

4.2.4 Recall. Recall is used to measure how often the model can correctly classify a molecule as toxic from the set of molecules that it classified as toxic.

4.2.5 F1-Score. The F1-score is used to measure the harmonic mean between precision and recall.

4.3 Experiments

4.3.1 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. It is performed on the testing dataset, and represents the average error upon observing completely new data. The results can be seen in table 1.

Model	Validation Loss	Test Loss	Accuracy
GCN	0.2109	0.2300	0.7565
GAT	0.2097	0.2252	0.7622

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

Overall, the GAT model performed more accurately than the GCN model. This was in line with our predictions because specific bonds or functional groups can have a greater impact on toxicity. This means that in theory, weighting all atoms the same as in the GCN model would not accurately reflect the real-world logic for toxicity mechanisms [12], and thus would not perform as well as the GAT model.

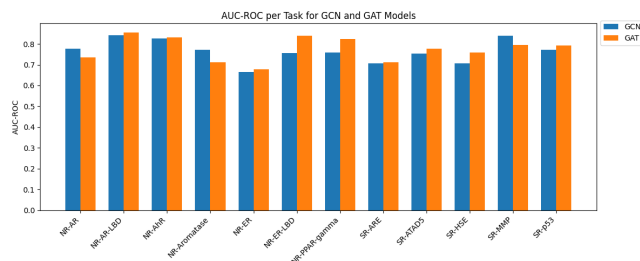


Figure 4: AUC-ROC per toxicity target for GCN and GAT

As mentioned in section 3.1, the Tox21 dataset also tells us which of the 12 toxicity targets are activated by a molecule. To gain a deeper understanding of how well the two GNN models perform in accurately classifying these targets, we calculated and plotted the AUC-ROC of each toxicity target for both models. The results from

the plot can be observed in figure 4. The most important conclusion to draw from the plot is that the GAT model does not have an AUC-ROC score higher than that of GCN for all toxicity targets. The targets where GCN scored higher are as follows:

- NR-AR
- NR-Aromatase
- SR-MMP

4.3.2 Experiment 2. Building on the previous experiment, we continued to compare the GNN models, but this time we wanted to dive deeper into the biological/chemical interpretation of our data. To accomplish this, we focused on the top 10 molecules where the models disagreed the most with their predictions. Disagreements were measured by the absolute difference between their confidence in the predictions. To visualize the molecules, we also colour-coded them based on how much the atoms or bonds influence the GNN model’s prediction. For GATs, this would be based on the attention score. GCNs do not have attention scores, so we used saliency scores instead. Saliency is measured by computing how much a model’s output would change if you vary the feature vector of a node x_i . The full equation for computing saliency can be seen below in equation (3).

$$\text{saliency}_i = \|\nabla_{x_i} \text{prediction}\| \quad (3)$$

Atoms/bonds with high attention or saliency scores were highlighted in red when drawing the molecules.

Index: 621
 SMILES: CO[C@H]1C[C@H](O[C@H]2[C@H](C)C(=O)O[C@H](C)[C@H](C)C[C@H](N(C)C)[C@H]3OC(C=O)[C@H]2C)O[C@H]1(C)[C@H]3O
 GCN_pred: 0.214
 GAT_pred: 0.933
 abs_diff: 0.181
 Correct model: GAT
 Tasks disagreeing: ['NR-AR', 'NR-AR-LBD']

GAT Attention:

GCN Saliency:

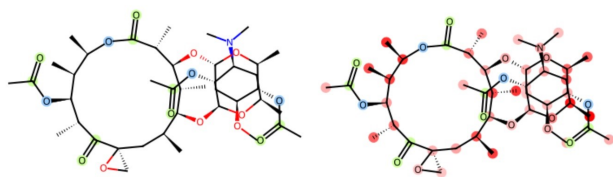


Figure 5: GAT attention vs. GCN saliency for molecule 621

In figure 5, molecule 621 is used as an example where the GAT model was correct, and non-toxic. In the GAT attention diagram, oxygen bonds are specifically highlighted, and the model outlines substructures more clearly. In the GCN saliency diagram, the highlighted atoms seem more random, as many of them are highlighted but the model’s prediction turned out to be incorrect. The results from this molecule demonstrate how the GAT model can identify important substructures to help with its predictions, something that the GCN model is incapable of.

Index: 1
 SMILES: CCCC[Sn](CCCC)(CCCC)CCCC
 GCN_pred: 0.123
 GAT_pred: 0.449
 abs_diff: 0.346
 Correct model: GCN
 Tasks disagreeing: ['NR-AR-LBD', 'NR-ER-LBD', 'NR-PPAR-gamma', 'SR-ARE', 'SR-ATADS', 'SR-HSE', 'SR-MMP', 'SR-p53']

GAT Attention:

GCN Saliency:

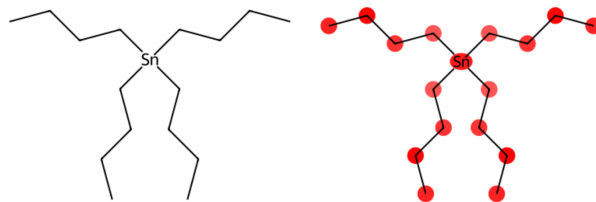


Figure 6: GAT attention vs. GCN saliency for molecule 1

In figure 6, molecule 1 is used as an example where the GCN model was correct, and non-toxic. In the GAT attention diagram, none of the atoms or bonds is highlighted in red, meaning that they do not have high attention scores to contribute to its prediction. In the GCN saliency diagram, all the atoms are highlighted in red, meaning they contribute greatly to its prediction. Because the GCN model was the correct one, this example shows the advantage GCNs have over GATs at predicting toxicity for molecules that do not contain any substructures. The full results from this experiment conducted on a total of the top 10 disagreeing molecules can be found in appendix B.3 of this report.

4.3.3 Experiment 3. The third experiment focuses solely on the GAT model, and it looks for how enriched certain substructures known as toxicophores are in the top-attention molecules. We obtained the set of toxicophores SMARTS used from the ToxAlerts web server by Sushko et. al (2012) [13]. SMARTS are patterns to search for inside a molecule’s SMILES string. The full list of toxicophore SMARTS used in this experiment are given below in table 2.

Toxicophore Name	SMARTS Pattern
Nitro Group	[N+](=O)[O-]
Aromatic Amine	[NX3;H2,H1;!\$(NC=O)]
Halogen	[F,C1,Br,I]
Hydroxyl	[OX2H]
Carboxylic Acid	C(=O)[OH]
Ester	C(=O)O
Carbonyl	[CX3]=O
Amide	C(=O)N
Phenol	c[OX2H]
Aniline	c[NX3H2,NX3H,NX3]

Table 2: Table of toxicophore SMARTS used for enrichment

Since we wanted to focus on the chemical substructures the GAT model believes are most indicative of toxicity, we conducted this experiment on the top-attention molecules, specifically the top 1%, 5%, 10%, and 20% of molecules with the highest attention scores. We then calculated the enrichment of each toxicophore in the chosen

ranges. The formula used for calculating enrichment can be seen below in equation (4).

$$\text{Enrichment} = \frac{\text{freq}_{\text{top-attention}}}{\text{freq}_{\text{all molecules}}} \quad (4)$$

The enrichment scores for each toxicophore substructure were plotted on the heatmap seen in figure 7. Columns represented the percentage of top-attention molecules according to the GAT model, and rows represented the toxicophore substructure. Enrichment values less than 1 mean that the model pays less attention to these molecules, while values equal to 1 mean that the pattern occurs as often in top-attention molecules as in baseline, and values greater than 1 mean that the substructure is enriched so the model pays extra attention to it.

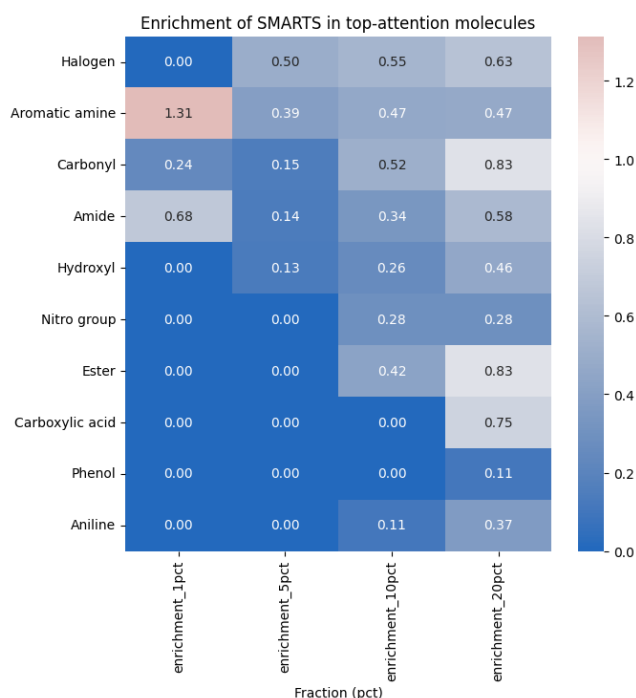


Figure 7: Heatmap showing the enrichment of toxicophore substructures in top-attention molecules

From the heatmap in figure 7, aromatic amine at 1% of top-attention molecules has the highest enrichment score with a value of 1.31. However, its enrichment value also significantly drops as the percentage of top-attention molecules increases. This suggests that the influence this substructure has on the model is concentrated only at extremely high attention values, and is not considered important otherwise. Ester and carboxylic acid groups have high enrichment values at 20% of top-attention molecules with enrichment scores of 0.83 and 0.75 respectively, which means these substructures are prevalent at high attentions. The carbonyl group’s enrichment score steadily increases as the percentage of molecules increases, so it is a very important functional group for the GAT’s attention.

4.3.4 Experiment 4. The fourth experiment will compare the performance of the baseline model to the GNNs from experiment 1. Each model is run separately on the same data. An error analysis is performed during all experiments, as well as standard machine learning metrics such as ROC AUC, Accuracy, Precision, Recall, and F1-score. The metrics are calculated per label of the 12 labels for all models. Table 3 shows the average metrics of the GNN models in comparison with the baseline RFC.

Model	AUC-ROC	Accuracy	Precision	Recall	F1-Score
GCN	0.732384	0.756492	0.195589	0.516800	0.257348
GAT	0.751418	0.762239	0.189231	0.533661	0.260866
RandomForest	0.791187	0.620113	0.657729	0.250197	0.334193

Table 3: Experiment 4 example results: Comparing GNN models with RFC model

The results show that the three models have similar AUC-ROC ~0.7 which indicates that the models are competitive. The accuracy for the two GNN models is close ~0.75-0.76, with GAT slightly better than GCN. On the other hand, the RFC model has significantly lower accuracy at ~0.62. This corresponds with the recall scores ~0.5 and ~0.2 for the GNNs and RFC respectively. The precision of RFC is better compared to the GNNs, but this is less significant because the dataset is imbalanced. Since the F1-score depends on precision, RFC is also better, but not as much because of the low recall score.

5 Conclusions and Future Work

To recap the work we have done so far, we have created four experiments to test our model for predicting toxicity in molecules. The first three experiments involve implementing the GCN and GAT models, and comparing the accuracy and performance of the two models. The fourth experiment implements the baseline model, a Random Forest Classifier with ECFP representation of molecules. We then followed up on the results from experiment 1 by comparing the baseline model to the GNN models.

In the first experiment, the GCN model performed with an accuracy of 75.65% and the GAT model with 76.22%. As expected, the GAT model was more accurate, but the results are very close. In addition, GAT had better AUC ROC scores in all labels besides NR-AR, NR-Aromatase, and SR-MMP. In the fourth experiment, the GNN models achieved the expected behavior of higher accuracy compared to the baseline model.

Given these results, we suggest that an ideal GNN model should include both GCN and GAT models, where the GCN will be responsible for predicting the NR-AR, NR-Aromatase, and SR-MMP labels, and the GAT for the rest of the targets.

Due to time constraints, we did not get to fine-tune the hyperparameters of the GNNs as originally planned to see if model accuracy could be improved. If this was implemented, it would be done through an automated, iterative process where different combinations of hyperparameters would be tested, and then plotted to see which version performed the best.

Addressing the imbalance in the dataset will also be important for future iterations. One approach that could be taken is integrating

data from other datasets such as ClinTox or ToxCast, increasing the number of molecules with non-toxic labels.

Further future improvements in the performance testing may 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, the dataset will be split into k folds before training each model, and then iteratively use each fold as a training set while the other $k - 1$ folds will be used as testing sets. A benefit of using this method is that you can analyze 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 Velićović, 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.
- [10] Yanli Liu, Yourong Wang, Jian Zhang, Baoxiang Liu, Maode Ma, and Jincai Chang. New machine learning algorithm: Random forest. pages 246–252, 2012.
- [11] Jin Zhang, Daniel Mucs, Ulf Norinder, and Fredrik Svensson. LightGBM: An Effective and Scalable Algorithm for Prediction of Chemical Toxicity—Application to the Tox21 and Mutagenicity Data Sets. *Journal of Chemical Information and Modeling*, 59(10):4150–4158, October 2019.
- [12] J D McKinney. The molecular basis of chemical toxicity. *Environmental Health Perspectives*, 61:5–10, 09 1985.
- [13] Iurii Sushko, Elena Salmina, Vladimir Potemkin, Gennadiy Poda, and Igor V Tetko. Toxalerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions. *Journal of Chemical Information and Modeling*, 52:2310–2316, 08 2012.

A Github Repository

Please visit <https://github.com/goIsaiah/ToxicityPredictor> for the full code implementation, and visualization tools used in this project.

B Supplementary Tables and Figures

B.1 Top 9 Attention Molecules

In figure 8, we plotted the top 9 molecules with the highest attention scores according to the GAT model to get a greater understanding of what the GAT pays attention to. After conducting more experiments however, we realized that the information here was not as helpful as the data collected from experiment 2 in section 4.3.2.

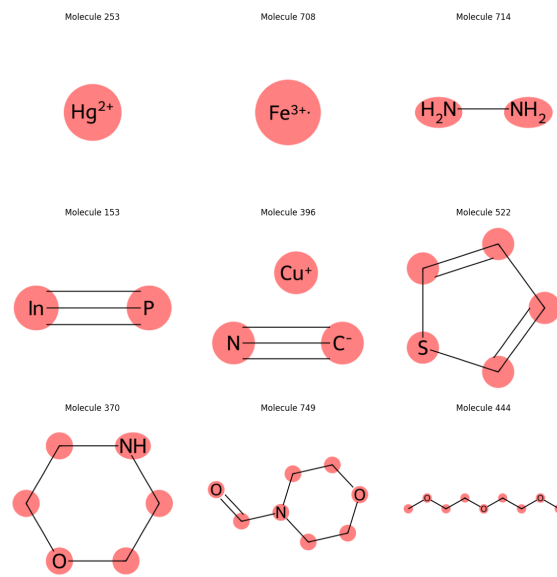


Figure 8: Top 10 molecules with the highest attention scores

B.2 Enrichment With Overlap Table

The following data in table 4 was used to help create the heatmap in figure 7 for experiment 3 in section 4.3.3 of this report. The heatmap was a more helpful way to visually convey the same information.

Toxicophore	Background Count	Top k count	Overlap Fraction
Carbonyl	402	67	0.426752
Ester	235	39	0.248408
Hydroxyl	303	28	0.178344
Halogen	197	25	0.159236
Amide	145	17	0.108280
Aniline	187	14	0.089172
Aromatic Amine	149	14	0.089172
Carboxylic Acid	87	13	0.082803
Nitro Group	35	2	0.012739
Phenol	89	2	0.012739

Table 4: Enrichment with overlap table

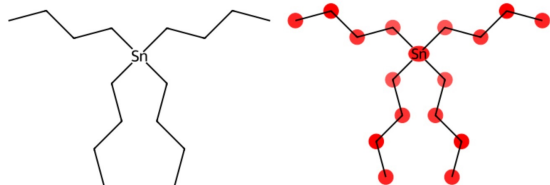
B.3 Molecules With the Biggest Disagreements

Figure 9 shows all 10 molecules we tested for experiment 2 in section 4.3.2 of this report.

Index: 1
 SMILES: CCCC[Sn](CCCC)(CCCC)CCCC
 GCN_pred: 0.123
 GAT_pred: 0.440
 abs_diff: 0.346
 Correct model: GCN
 Tasks disagreeing: ['NR-AR-LBD', 'NR-ER-LBD', 'NR-PPAR-gamma', 'SR-ARE', 'SR-ATAD5', 'SR-HSE', 'SR-MWP', 'SR-p53']

GAT Attention:

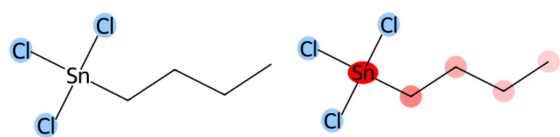
GCN Saliency:



Index: 2
 SMILES: CCCC[Sn](Cl)(Cl)Cl
 GCN_pred: 0.156
 GAT_pred: 0.358
 abs_diff: 0.226
 Correct model: GCN
 Tasks disagreeing: ['SR-ARE', 'SR-HSE', 'SR-MWP']

GAT Attention:

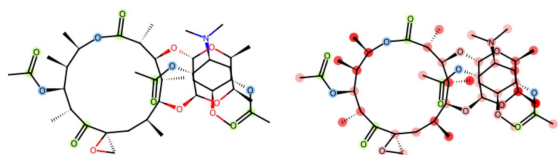
GCN Saliency:



Index: 621
 SMILES: CO[C@H]1C[C@H](O[C@H]2[C@H](C(C=O)O[C@H](C)[C@H](C)[C@H](OC(C)=O)[C@H](C)C(=O)[C@H]3(CO)C[C@H](C)[C@H](O[C@H]3O)[C@H](C)[C@H](N(C)C)[C@H]3OC(C)=O)[C@H]2C)O[C@H](C)[C@H]1OC(C)=O
 GCN_pred: 0.214
 GAT_pred: 0.033
 abs_diff: 0.181
 Correct model: GAT
 Tasks disagreeing: ['NR-AR', 'NR-AR-LBD']

GAT Attention:

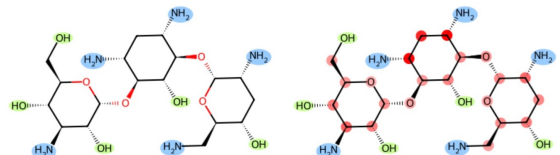
GCN Saliency:



Index: 185
 SMILES: NC[C@H]1O[C@H]2[C@H](O[C@H](N)[C@H](O[C@H]3O[C@H]3O[C@H](CO)[C@H](O)[C@H](N)[C@H]3O)[C@H]2O)[C@H](N)[C@H]1O
 GCN_pred: 0.235
 GAT_pred: 0.060
 abs_diff: 0.175
 Correct model: GAT
 Tasks disagreeing: ['NR-AR', 'NR-AR-LBD']

GAT Attention:

GCN Saliency:



Index: 253
 SMILES: [Hg+2]
 GCN_pred: 0.043
 GAT_pred: 0.203
 abs_diff: 0.160
 Correct model: Both
 Tasks disagreeing: []

GAT Attention:

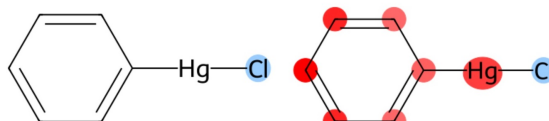
GCN Saliency:

Hg²⁺

Index: 558
 SMILES: Cl[Hg]Clcccc1
 GCN_pred: 0.305
 GAT_pred: 0.458
 abs_diff: 0.291
 Correct model: GCN
 Tasks disagreeing: ['NR-AR-LBD', 'NR-ER-LBD', 'NR-PPAR-gamma', 'SR-ARE', 'SR-ATAD5', 'SR-HSE', 'SR-MWP', 'SR-p53']

GAT Attention:

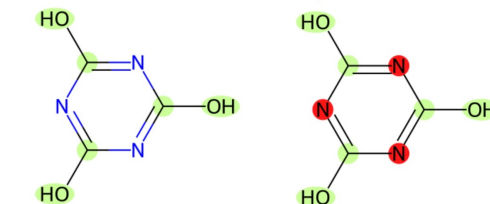
GCN Saliency:



Index: 719
 SMILES: Oc1nc(O)nc(O)n1
 GCN_pred: 0.093
 GAT_pred: 0.312
 abs_diff: 0.218
 Correct model: GCN
 Tasks disagreeing: ['NR-AHR', 'NR-ER', 'SR-ARE', 'SR-MWP']

GAT Attention:

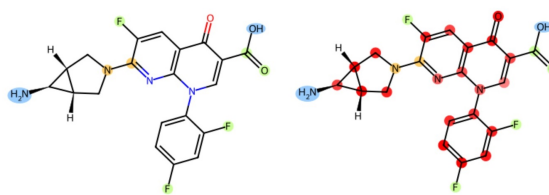
GCN Saliency:



Index: 316
 SMILES: N[C@H]1[C@H]2CN(C3C4C(Cc3F)c(=O)c(C(=O)O)c4-c3ccc(F)cc3F)[C@H]12
 GCN_pred: 0.304
 GAT_pred: 0.129
 abs_diff: 0.175
 Correct model: Both
 Tasks disagreeing: []

GAT Attention:

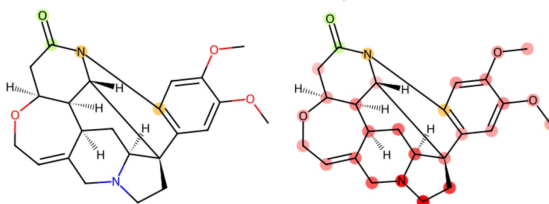
GCN Saliency:



Index: 26
 SMILES: COc1cc2c(cc1OC)[C@H]13CCNACC5=CCO[C@H]6CC(=O)N2[C@H]1[C@H]6[C@H]5C[C@H]43
 GCN_pred: 0.223
 GAT_pred: 0.061
 abs_diff: 0.161
 Correct model: Neither
 Tasks disagreeing: []

GAT Attention:

GCN Saliency:



Index: 620
 SMILES: CC(=O)O[C@H]1C[C@H]2CC[C@H]3[C@H](CC[C@H]4(C)[C@H]3C[C@H]([N+](=O)3CCCC3)[C@H]4OC(C)=O)[C@H]2(C)C[C@H]1[N+](=O)C)CC1
 GCN_pred: 0.234
 GAT_pred: 0.064
 abs_diff: 0.157
 Correct model: GAT
 Tasks disagreeing: ['NR-AR', 'NR-ER']

GAT Attention:

GCN Saliency:

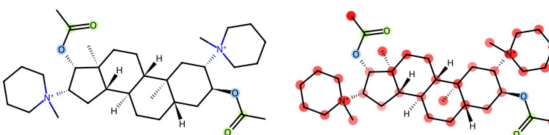


Figure 9: Full diagram of the top 10 molecules where the GNN models disagree the most