Graph Neural Networks : CS 7643

Robert Adams
Georgia Institute of Technology
radams71@gatech.edu

Raymond Wu
Georgia Institute of Technology
rwu366@gatech.edu

Pedram Gharani
Georgia Institute of Technology
pgharani3@gatech.edu

Abstract

Graph Neural Networks are used to classify data that has a specific relational structure such as social media connections or molecular data. We offer a thorough analysis of GNNs used over a molecular benchmark dataset, by creating our own custom architecture and evaluating a modern architecture DeeperGCN. We also explore solutions to severely imbalanced datasets through focal loss and AUC margin loss. Our experiments are documented within this report which thoroughly explore the aspects of our custom architecture, the affects of aggregation functions, and how different loss functions help optimize the ROC-AUC metric.

1. Introduction/Background/Motivation

Geometric Deep Learning offers a framework for creating neural network architectures which effectively exploit the symmetries of the data set. Graph Neural Networks (GNNs) are one particular type of architecture which a designed to work for inputs which are graphs.

A graph $G = (V, E)$ consists of a set of vertices or nodes V and a set of edges $E \subseteq V \times V$ which connect pairs of vertices. A graph automorphism or symmetry is a permutation σ on the set of vertices V that preserves edge-vertex connectivity, i.e. $\forall u, v \in V, (u, v) \in E \iff (\sigma(u), \sigma(v)) \in E$. Graph neural networks aim to capture this inductive bias through aggregation functions and pooling layers.

Definition 1 (Aggregation Functions). Let $\mathcal{X}$ be the space of multisets over $\mathbb{R}^d$. We define an aggregation function as a function $\phi : \mathcal{X} \to \mathbb{R}^d$ which is permutation invariant, i.e. for any multiset $X \in \mathcal{X}$ and any permutation σ on X, we have $\phi(\sigma(X)) = \phi(X)$[3]

Analogous to Convolutional Networks, aggregation functions are used locally (on a vertex v and its neighborhood $\mathcal{N}_v$, i.e. all vertices connected to v by an edge) and are invariant under local symmetries. These functions are applied to features or messages from each vertex and combined to create a permutation equivariant operation ($\psi(\sigma(x)) = \sigma(\psi(x))$ similar to how a classical convolution layer is built.

In this paper, we focus on the task of graph property prediction, i.e. a classification problem on the whole graph. Link prediction (predicting edges between vertices) and Node classification are two other well-known tasks that graph neural networks can accomplish. The architectures described in this paper are easily adaptable to these tasks; however, we will not explore these tasks in this paper. A key difference is for graph prediction, our GNN will include a global pooling layer which is just an aggregation function that is applied globally to the whole graph.

Predicting properties of chemical compounds is a common application of graph property prediction. We aimed to explore several different architectures of GNNs, experimenting with different aggregation functions, while also experimenting with some more general aspects of deep learning such as different loss functions.

DeeperGCN [5] is a modern architecture for GNNs which utilizes residual connections and learnable aggregation functions. Our ambitions included to better understand this architecture, to conduct experiments with elements of this architecture, and attempting to improve results using this architecture with more specialized loss functions and balancing the dataset. Another modern approach to graph problems is using transformers. Graphormer [9] using specialized encoding methods to help the transformer capture the structural nature of the graphs.

GNNs have lots of applications benefiting many areas of research and industry. Graphs can be used to represent data such as social networks and molecules. Any improvement in this are can lead to better recommendation systems in so-

cial media or can help advance medicine through identifying potentially useful chemical compounds. For the particular dataset we are working over, a well performing model can help identify molecules useful for HIV treatments, saving time and money for medical researchers.

We used *ogbg-molhiv*, a dataset adapted from MOLECULENET, created by Open Graph Benchmarks [2]. *Ogbg-molhiv* consists of 40,000+ molecules which are classified based on their ability to inhibit HIV replication (active or inactive). The dataset is extremely imbalanced with only 3.51% active molecules. The molecules are represented by graphs where nodes correspond to atoms and edges are chemical bonds. Each node is represented by a 9-dimensional feature vector consisting of properties such as atomic number, chirality, formal charge, etc. The edge features are 3 dimensional, consisting of bond type, bond stereochemistry, and whether the bond is conjugated. Training, validation, and testing sets (80/10/10 split) are created using scaffolding splitting which splits the molecules based on their two-dimensional structure placing structurally different molecules into different subsets.

The metric used for evaluation is ROC-AUC or the area under the receiver operating characteristic curve. The ROC curve shows the True Positive Rate ($\frac{TP}{TP+FN}$) against the False Positive Rate ($\frac{FP}{FP+TN}$) at every possible threshold for a given binary classifier. The area under this curve represents the probability that a randomly chosen positive example is ranked higher by the classifier than a randomly chosen negative example [1]. ROC-AUC offers a reasonable metric for our dataset as it is not very susceptible to data imbalance. However, one downside is ROC-AUC does not care about the magnitude of the output probabilities of the model, but rather it only captures the an evaluation of the ranking of positive examples over negative. For instance, a model which output 0.1 for all positive examples and 0.05 for all negative examples would achieve a perfect ROC-AUC score of 1.

SOTA performance on this dataset is achieved via hypergraph ensembling of several other well-performing models [8]. The testing and validation ROC-AUC achieved are 0.8475 ± 0.0003 and 0.8275 ± 0.0008, respectively.

2. Approach

We used PyG (pytorch geometric) for implementation of Custom ConvGCN and the provided implementation of DeeperGCN in PyG from LibAUC. The visualizations were generated using RDkit, Captum, and NetowrkX.

2.1. Custom ConvGCN

We designed a custom Graph Convolutional Network (GCN) architecture that integrates enhanced feature embeddings, attention-based pooling, and residual connections.

The architecture includes several key components. First, we enhance input features using embedding layers to capture the categorical nature of atomic and bond properties. Node features are embedded into a higher-dimensional space (`atom_embed_dim=64`), allowing the model to learn more expressive atomic representations. Similarly, edge features are embedded into a 16-dimensional space (`edge_embed_dim=16`) to encode bond characteristics. These embeddings are inspired by strategies for handling categorical data in graph-based models [2], ensuring that the model can fully exploit the structural information present in the *ogbg-molhiv* dataset.

At the core, the model uses multiple graph convolutional layers implemented with PyTorch Geometric's `Conv` modules. These layers aggregate information from neighboring nodes to update node representations. A sequence of three convolutional layers with hidden dimensions of 192, 384, and 768 progressively captures higher-level graph features. Each layer uses a locally applied permutation-invariant aggregation function (e.g., *mean* aggregation), preserving the inherent symmetries of the graph structure, as defined in Definition 1 [3]. To improve gradient flow and training stability, residual connections are incorporated between layers, inspired by the DeeperGCN architecture [5]. The output of each convolutional layer is added to its input after a linear transformation, enabling the model to learn incremental updates to the node embeddings.

The final pooled representation is passed through a fully connected layer to produce the predicted probability of HIV inhibition. The model is trained using binary cross-entropy loss, but we also explored using focal loss to deal with the dataset imbalance. We used both the attention scores and predictions to create visualization shown in Section 3.2. Our custom GCN architecture combines enhanced embeddings, deep convolutional layers, attention-based grouping, and residual connections to effectively learn molecular representations for the *ogbg-molhiv* task.

2.2. Multi-headed Attention Pooling and Residual Connection

A key aspect of our custom GCN architecture is the integration of multi-headed attention pooling and residual connections, which enhance the model's ability to learn molecular representations. These components address the challenges of aggregating node-level features into a graph-level representation and ensuring stable training for deep graph convolutional layers.

The multi-headed attention pooling mechanism aggregates node representations into a single graph-level representation after the convolutional stack, which computes attention scores for each node based on its representation, allowing the model to focus on the most relevant nodes. We use 4 attention heads, with each head computing a scaled

dot-product attention score, followed by concatenation and a linear transformation to produce the final pooled representation. The attention mechanism is inspired by Graph Attention Networks [7], which use attention to weigh the importance of neighboring nodes, adapted here for global pooling. Mathematically, for a node representation $h_i \in \mathbb{R}^d$, the attention score for head k is computed as:

$$\alpha_{i,k} = \text{softmax}\left(\frac{(W_k h_i)^\top (W_k h_j)}{\sqrt{d_k}}\right), \tag{1}$$

where $W_k \in \mathbb{R}^{d_k \times d}$ is a learned projection matrix, $d_k = d/n$ where n is the number of heads, and the softmax is applied over all nodes j. The attention scores are also used for interpretability, as they indicate which atoms contribute most to the prediction.

To improve gradient flow and training stability, we incorporate residual connections between the graph convolutional layers, inspired by the DeeperGCN architecture [5]. For each convolutional layer, the output is added to its input after a linear transformation to match dimensions, i.e., $h^{(l+1)} = h^{(l)} + \text{Conv}(h^{(l)})$, where $h^{(l)}$ is the node representation at layer l. This design mitigates the vanishing gradient problem, enabling the model to learn incremental updates to node representations across the deep stack of layers (dimensions 192, 384, and 768). We also apply dropout (`dropout=0.2`) within the convolutional and attention layers to prevent overfitting, a choice informed by the hyperparameter settings.

The combination of multi-headed attention pooling and residual connections allows our GCN to effectively capture both local and global patterns in molecular graphs while maintaining training stability. The attention mechanism enhances interpretability by highlighting key substructures, while the residual connections ensure that the model can scale to deeper architectures without sacrificing performance.

2.3. DeeperGCN

DeeperGCN is an architecture that allows for training deeper networks via residual connections that help solve the vanishing gradient problem. DeeperGCN also introduces the idea of *generalized aggregation functions* which are defined as a continuous family of aggregation functions ϕ_θ for some learnable parameter θ. One such example is the *generalized softmax* which is defined as:

$$\sum_{u \in \mathcal{N}_v} \frac{e^{\beta m_{vu}}}{\sum_{i \in \mathcal{N}_v} e^{\beta m_{vi}}} m_{vu} [5] \tag{2}$$

where m_{vu} represents the combined message or feature from the vertex v, the vertex u, and the edge (u, v). A nice property of this function is that as $\beta \to 0$, the coefficients go to $\frac{1}{|\mathcal{N}_v|}$ and thus we get the *mean* aggregation. As $\beta \to \infty$, the coefficient go to 1 if m_{vu} is a maximal element and 0 otherwise, hence we get the *max* aggregation. We can interpret this generalized function as a learnable interpolation of *mean* and *max*.

Originally, we set out to experiment with new aggregation functions for the DeeperGCN architecture. However, we were not able to implement a novel generalized function. We chose to focus more on tuning this architecture with the use of sampling and ROC-AUC specific loss functions.

Additionally, we took the implementation of Random Forest classifications on chemical fingerprints generated from the molecules from [11]. These predictions were used for tuning models for slightly increased performance.

2.4. AUCMLoss and Sampling

We experimented with LibAUC's implementation of AUC margin loss which aims to better optimize the ROC-AUC metric. The objective function is defined below:

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^d \\ (a,b) \in \mathbb{R}^2}} \max_{\alpha \in \mathbb{R}^+} f(\mathbf{w}, a, b, \alpha), \tag{3}$$

$$\begin{aligned} f(\mathbf{w}, a, b, \alpha) = & \mathop{\mathbb{E}}_{y=1}\left[(h_{\mathbf{w}}(x) - a)^2\right] + \mathop{\mathbb{E}}_{y=-1}\left[(h_{\mathbf{w}}(x) - b)^2\right] \\ & + 2\alpha\left(m + \mathop{\mathbb{E}}_{y=-1}[h_{\mathbf{w}}(x)] - \mathop{\mathbb{E}}_{y=1}[h_{\mathbf{w}}(x)]\right) \\ & - \alpha^2 [12] \end{aligned} \tag{4}$$

$h_{\mathbf{w}}$ represents the model with input x and label y. Details for how this objective function maximizes ROC-AUC are beyond the scope of this paper and can be found here [10].

We also used LibAUC's AUC composition loss which is a composition of AUC margin loss with cross entropy, defined by $L_{AUC}(\mathbf{w} - \alpha \nabla L_{CE}(\mathbf{w}))$, where α is the inner step size for the cross entropy updates.

We aimed to explore how resampling and how different loss functions compare with different imbalance ratios on the data. We tested different sampling rates (percentage of positive examples per batch) on the DeeperGCN architecture, and compared results with cross entropy loss, AUC margin loss, and AUC composition loss.

3. Experiments and Results

3.1. Custom GCN Model Results

We evaluated our custom GCN model, described in Section 2.1, on the *ogbg-molhiv* dataset.

We trained our custom GCN using binary cross-entropy and focal loss for 100 epochs, with a batch size of 128 and a learning rate of 0.001, using the Adam optimizer. The model leverages enhanced embeddings

(`atom_embed_dim=64`, `edge_embed_dim=16`), a stack of three convolutional layers (dimensions 192, 384, 768), residual connections, and multi-headed attention pooling, as detailed in Sections 2.1 and 2.2.

Our custom GCN achieved a test ROC-AUC of 0.785 and a validation ROC-AUC of 0.823, demonstrating decent performance compared to baseline GCN models but falling significantly short of the state-of-the-art. The use of residual connections improved training stability, allowing the model to scale to deeper layers without suffering from vanishing gradients, while the multi-headed attention pooling enabled the model to focus on relevant molecular substructures, as explored in Section 3.2. These results highlight the effectiveness of our architectural enhancements, though further improvements in handling dataset imbalance and optimizing the attention mechanism could narrow the gap with state-of-the-art methods.

Table 2 shows results for our custom model with several different aggregation functions and compares them to a DeeperGCN model.

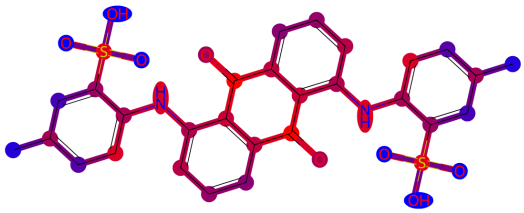
3.2. Interpreting Model Predictions through Attention Visualization

To interpret the decisions made by our custom ConvGCN model, we developed a visualization pipeline to highlight which parts of a molecular graph the model focuses on when making predictions. Specifically, we visualize attention weights from the `MultiHeadAttention` layer on 2D molecular structures using RDKit, a cheminformatics toolkit. Each atom and bond is colored from blue (low attention) to red (high attention), based on normalized attention scores. The bond attention is computed as the average attention of its two adjacent atoms.

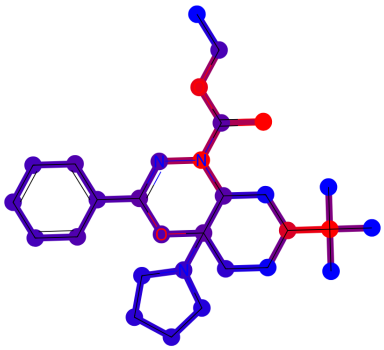
Our visualization approach draws on prior work in GNN interpretability [7, 6] and uses rendering techniques adapted from RDKit tutorials [4].

We present visualizations for four representative test-set molecules, selected to reflect a range of prediction outcomes (Figure 1):

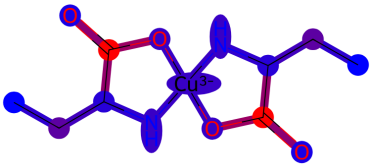
- **True Positive** ($y_{\text{true}} = 1$, $y_{\text{pred}} = 0.863$): The model confidently identifies this molecule as an HIV inhibitor, assigning high attention to a compact, central substructure, suggesting effective focus on relevant graph regions.
- **False Negative** ($y_{\text{true}} = 1$, $y_{\text{pred}} = 0.087$): Although the model assigns moderate attention to an informative-looking subgraph, it misclassifies the molecule, likely due to underrepresentation of similar structures during training.
- **True Negative** ($y_{\text{true}} = 0$, $y_{\text{pred}} = 0.009$): The model accurately predicts the molecule as inactive, with attention centered on a consistent structural pattern associated with negative class labels.



(a) True Positive



(b) False Negative



(c) True Negative

Figure 1: ConvGCN attention visualizations on test molecules. Attention weights are visualized using a red (high) to blue (low) color scale.

These visualizations demonstrate that ConvGCN can learn to focus attention on chemically meaningful substructures without explicit domain knowledge. While successful predictions often correspond with focused attention on informative regions, failure cases reveal limitations—such as under-attending to rare or complex subgraphs, including metal atoms or sparse functional groups.

Conclusion and Implications. This qualitative analysis confirms that attention mechanisms in GNNs offer interpretable signals about which parts of the graph drive predictions. Moreover, attention visualization can highlight limitations in generalization due to training data sparsity. Future improvements may include data augmentation for rare patterns, integration of domain-specific features (e.g., 3D spatial data), or loss function modifications to boost recall on minority classes.

3.3. DeeperGCN and AUCMloss

We trained several DeeperGCN models experimenting with different parameters, and testing out AUC Margin loss and Compositional AUC loss. AUCM loss and Compositional AUC loss both require a positive example in each batch to be computed, so it is recommended to resample the dataset to control the number of positive examples. This is beneficial anyway as our dataset is severely imbalance, and while this doesn't affect ROC-AUC, it still affects cross entropy loss which makes it more difficult for the model to learn. Table 1 shows a comparison of models trained with each loss function versus the sampling rate. We can see the AUCM loss achieves a much better score on imbalanced data than cross entropy. As the data set becomes more balanced, we see the loss functions achieve much more similar scores. As cross entropy does nothing to combat the data imbalance and AUCM is much less susceptible to imblanace, this behavior is not surprising.

These experiments were conducted over 100 epochs, using Adam optimizer for cross entropy, and the appropriate LibAUC optimizers for the AUCM and compositional AUC objectives (for a full list of parameters see the associated github repository). We can also see in the loss and ROC-AUC curves in Figure 2 that cross entropy loss continues to decrease even as ROC-AUC has plateaued. As cross entropy care about the magnitude of the outputs and this experiment was conducted with a 30% sample rate, the cross entropy loss can decrease without affecting the overall ranking of samples, and thus leaving the ROC-AUC unchanged.

3.4. Visualizations via Captum

Visualizations on a molecule which is positively labeled were generated via Captum using their Integrated Gradients method and are shown in Figure 3. A 3-layer basic GNN was trained with a variety of different aggregation functions and were compared to the DeeperGCN model. Integrated gradients computes which features of the input data that are most important to the output of the network by using numerical integration and comparison to a baseline input. Higher values are represented by darker colors and wider edges which correspond to greater importance. These visualizations could help identify sub-structures in chemical compounds which are important for HIV inhibition.

We can see the model chooses to focus on drastically different parts of the molecule based on the aggregation function. We see a similarity between the simple GNN with *softmax* and the DeeperGCN molecule, as both are very center focused, and we can see that relative to the other models the *softmax* gives one of the highest prediction probabilities. For the *product* aggregation, we see very low importance for the majority of nodes and edges which can be explained by products exponentially blowing up and shrinking large and small numbers.

	sr=5%	sr=10%	sr=30%	sr=50%
AUCM	0.8094	0.8163	0.8390	0.8158
CE	0.7797	0.7991	0.8105	0.8084
CompAUC	0.8012	0.8118	0.8208	0.8311

Table 1: Validation ROC-AUC for DeeperGCN trained with different loss functions.

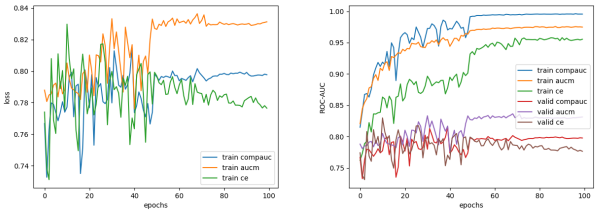


Figure 2: Training loss curves and ROCAUC curves for DeeperGCN trained with different loss functions.

4. Conclusion and Future Work

4.1. Project Success

This project aimed to predict HIV inhibition properties of molecules using graph neural networks (GNNs) on the *ogbg-molhiv* dataset. We evaluated two models: a baseline DeeperGCN model with AUC-optimized loss, and a custom-designed ConvGCN architecture.

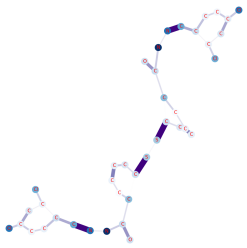
Our custom ConvGCN model achieved a test AUC of 0.7846 using *sum* aggregation which is significantly lower than SOTA performance and the performance we achieved with DeeperGCN. Originally, we had hoped to implement custom aggregation functions with DeeperGCN, and use these and other techniques of dealing with the imbalanced data to improve the baseline model for DeeperGCN AUC optimization of 0.8352±0.0054 Test ROC-AUC and 0.8238±0.0061 Validation ROC-AUC. However, we did not achieve this goal and focused more on existing implementations.

4.2. Future Work

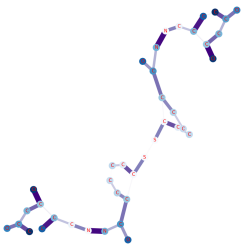
To improve the performance of our custom model, we would like to further explore the following areas:

1. **Dataset Augmentation:** Increase the number of positive examples through data augmentation or external sources to mitigate class imbalance without relying solely on oversampling techniques.

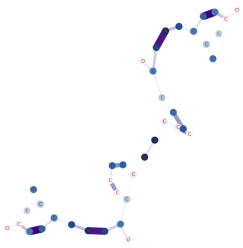
2. **Architectural Exploration:** Investigate hybrid models that combine the strengths of ConvGCN (local substructure focus) and DeeperGCN (global representation capacity), or explore deeper variants of ConvGCN to improve expressive power.



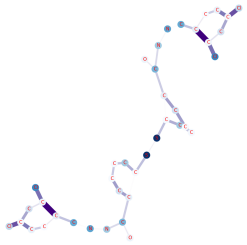
(a) *Max*
Valid ROC-AUC: 0.8074
Model Prediction: 0.0190



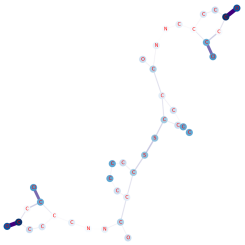
(b) *Mean*
Valid ROC-AUC: 0.7713
Model Prediction: 0.0560



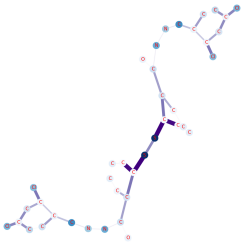
(c) *Median*
Valid ROC-AUC: 0.7552
Model Prediction: 0.1097



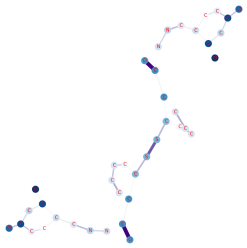
(d) *Min*
Valid ROC-AUC: 0.7977
Model Prediction: 0.0581



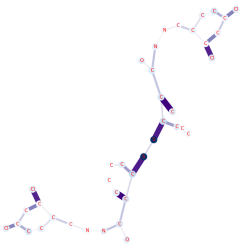
(e) *Product*
Valid ROC-AUC: 0.7822
Model Prediction: 0.0509



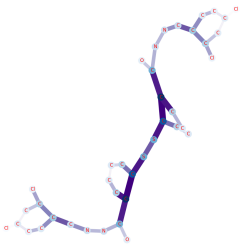
(f) *Softmax*
Valid ROC-AUC: 0.7988
Model Prediction: 0.0922



(g) *Standard deviation*
Valid ROC-AUC: 0.8154
Model Prediction: 0.0341



(h) *Sum*
Valid ROC-AUC: 0.8025
Model Prediction: 0.0524



(i) DeeperGCN with *generalized softmax*
Valid ROC-AUC: 0.8345
Model prediction: 0.9374

Figure 3: Visualizations of Captum Integrated Gradient analysis on a HIV inhibiting molecule with different aggregation functions. The validation ROC-AUC scores are included for each model, as well as the model prediction on each molecule.

Model	**Test AUC**	**Valid AUC**	**Train Loss**	**Notes**
Custom ConvGCN	0.7846	0.8229	0.3043–0.3202	*sum* aggregation
Custom ConvGCN	0.7553	0.8213	0.3131–0.3276	*mean* aggregation
Custom ConvGCN	0.7642	0.8009	0.2995–0.3220	*max* aggregation
DeeperGCN	0.8140	0.8384	0.1365 (AUCM)	tuned using RF predictions

Table 2: Comparison of aggregation methods in ConvGCN models and DeeperGCN

5. Work Division

Summary of individual contribution is in Table 3

References

[1] David J. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine Learning*, 77:103–123, 2009. 2

[2] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021. 2

[3] Ryan Kortvelesy, Steven Morad, and Amanda Prorok. Generalised f-mean aggregation for graph neural networks, 2023. 1, 2

[4] Greg Landrum. Rdkit: Open-source cheminformatics software, 2023. 4

[5] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns, 2020. 1, 2, 3

[6] Paul E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10772–10781, 2019. 4

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. 3, 4

[8] Yue Gao Xinwei Zhang, Yifan Feng. Multi-model ensemble on hypergraph, 2024. 2

[9] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021. 1

[10] Yiming Ying, Longyin Wen, and Siwei Lyu. Stochastic online auc maximization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 3

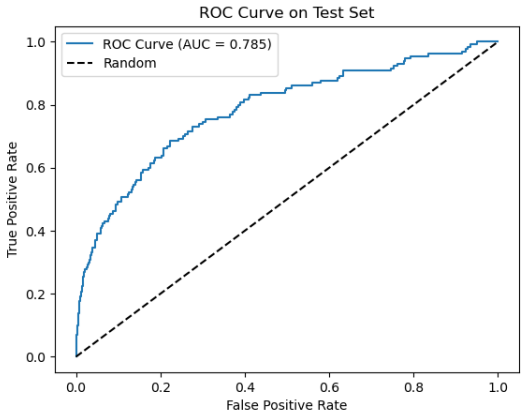
[11] Zhuoning Yuan, Yan Yan, Milan Sonka, and Tianbao Yang. Large-scale robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 3

[12] Zhuoning Yuan, Dixian Zhu, Zi-Hao Qiu, Gang Li, Xuanhui Wang, and Tianbao Yang. Libauc: A deep learning library for x-risk optimization. In *29th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023. 3

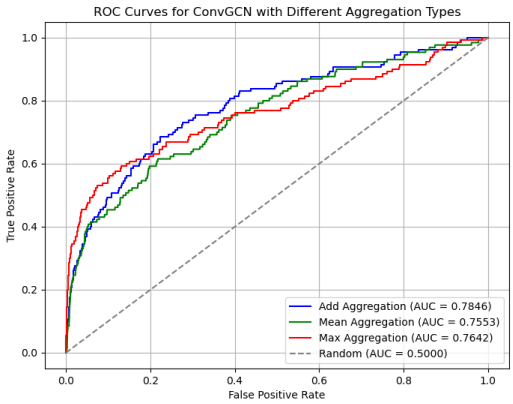
Student Name	Contributed Aspect	Details
Robert Adams	Implementation, Analysis, Tuning, and Visualizations	Implementation and experimentation with DeeperGCN and AUC losses. Generated Visualizatons via Captum Analysis of DeeperGCN, AUCM, and visualizations
Raymond Wu		
Pedram Gharani	Implementation, Analysis, Tuning, Visualization and Writing	Implementation and experimentation with the custom GCN model. Analysis of results and visualizations of attention mechanism.

Table 3: Contributions of team members.

A. ROC Curves



(a)



(b)

Figure 4: ROC Curve on TEST set for different aggregation types for Custom ConvGCN

B. Training Curves

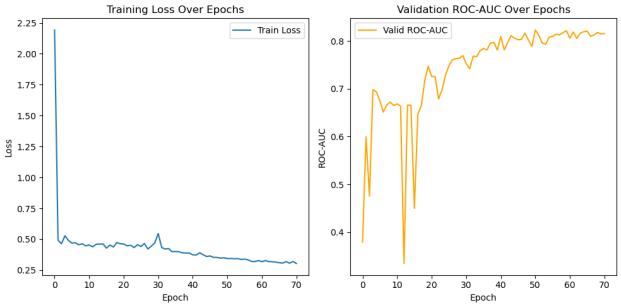


Figure 5: Training Metrics for Custom ConvGCN

C. Extra Visualizations

True Negative (Low Confidence): A second inactive molecule is correctly predicted, with attention focused on linker regions and minimal emphasis on surrounding atoms, highlighting reliance on localized structural features.

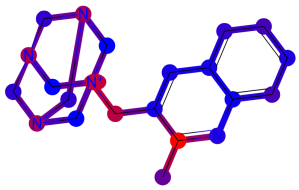


Figure 6: Caption