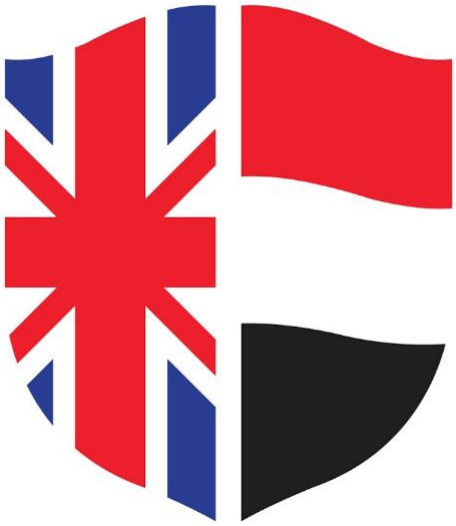

24CSCI33H

Artificial Neural Networks

Phase 2 Report

Subreddit Classification Using Graph Neural Networks

ID	Name	Email
224030	Amir Mohamed	amir224030@bue.edu.eg
226392	Merihan Ahmed	merihan226392@bue.edu.eg
228248	Omar Emara	omar228248@bue.edu.eg
218767	Sameh Ayman	sameh218767@bue.edu.eg

Table of Contents

Overview

This project aspires to evaluate different Graph Neural Network models including GraphSAGE, GCN, GIN, and GGNN on classifying post categories using the Reddit dataset. In Reddit, each post is authored under a subreddit, a category that groups together the posts discussing a specific topic. Gathered posts date back to the period between 2006 and 2016, and the dataset has a total of 3,848,330 instances. Each instance represents a post and its details, including the content, subreddit, author, etc.

Data Preprocessing

Four of the dataset features, which are the post id, author, post content, and the subreddit were selected to carry out the classification task. Firstly, post content undergone cleaning process where the special characters, extra spaces, stop words, and other irrelevant potential noise sources were removed. Secondly, the cleaned text was then embedded into vectors of numbers using Microsoft's "all-MiniLM-L6-v2" sentence-transformer which maps sentences and paragraphs into a 384-dimensional dense vector space. Thirdly, author, id, and subreddit features were all encoded into numbers. Lastly, other irrelevant features were dropped from the data frame making it ready for tackling the next step.

	author	body	normalizedBody	subreddit	subreddit_id	id	content	summary
0	raysofdarkmatter	I think it should be fixed on either UTC stand...	I think it should be fixed on either UTC stand...	math	t5_2qh0n	c69al3r	I think it should be fixed on either UTC stand...	Shifting seasonal time is no longer worth it.
1	Stork13	Art is about the hardest thing to categorize i...	Art is about the hardest thing to categorize i...	funny	t5_2qh33	c6a9nxd	Art is about the hardest thing to categorize i...	Personal opinions \n shit.
2	Cloud_dreamer	Ask me what I think about the Wall Street Jour...	Ask me what I think about the Wall Street Jour...	Borderlands	t5_2r8cd	c6acx4l	Ask me what I think about the Wall Street Jour...	insults and slack ass insight. \n Wall Street ...
3	NightlyReaper	In Mechwarrior Online, I have begun to use a m...	In Mechwarrior Online, I have begun to use a m...	gamingpc	t5_2sq2y	c8onqew	In Mechwarrior Online, I have begun to use a m...	Yes, Joysticks in modern games have apparently...
4	NuffZetPand0ra	You are talking about the Charsi imbue, right?...	You are talking about the Charsi imbue, right?...	Diablo	t5_2qore	c6acxvc	You are talking about the Charsi imbue, right?...	Class only items dropped from high-lvl monsters.

Table 1: first five instances of the dataset before preprocessing

	author	subreddit	id	content
0	98624	0	159019	[-0.005229626, 0.008069046, 0.010444595, -0.04...
1	144173	0	62385	[-0.06444688, -0.07142516, 0.024431113, -0.045...
2	89293	0	142816	[0.050086915, -0.0501369, 0.0778591, -0.013438...
3	160526	0	97433	[-0.07725685, -0.008193296, 0.04569599, 0.0350...
4	86037	0	43145	[-0.017911892, -0.021176245, 0.042304642, 0.02...

Table 2: first five instances of the data after preprocessing.

Data Representation

During this stage the preprocessed data was represented as a heterogenous graph where nodes represent authors and posts. Unweighted edges were created between authors' and their posts to indicate their ownership, and the weighted edges between posts represent their content cosine similarity. Due to the limited resources, only posts that had similarity equal to or greater than 90% were connected by an edge. PyTorch Geometric was employed to represent the data as a graph so that it can be fed to GNNs for carrying out the classification task. The following figure shows the representation graph of the first 15 posts of the data:

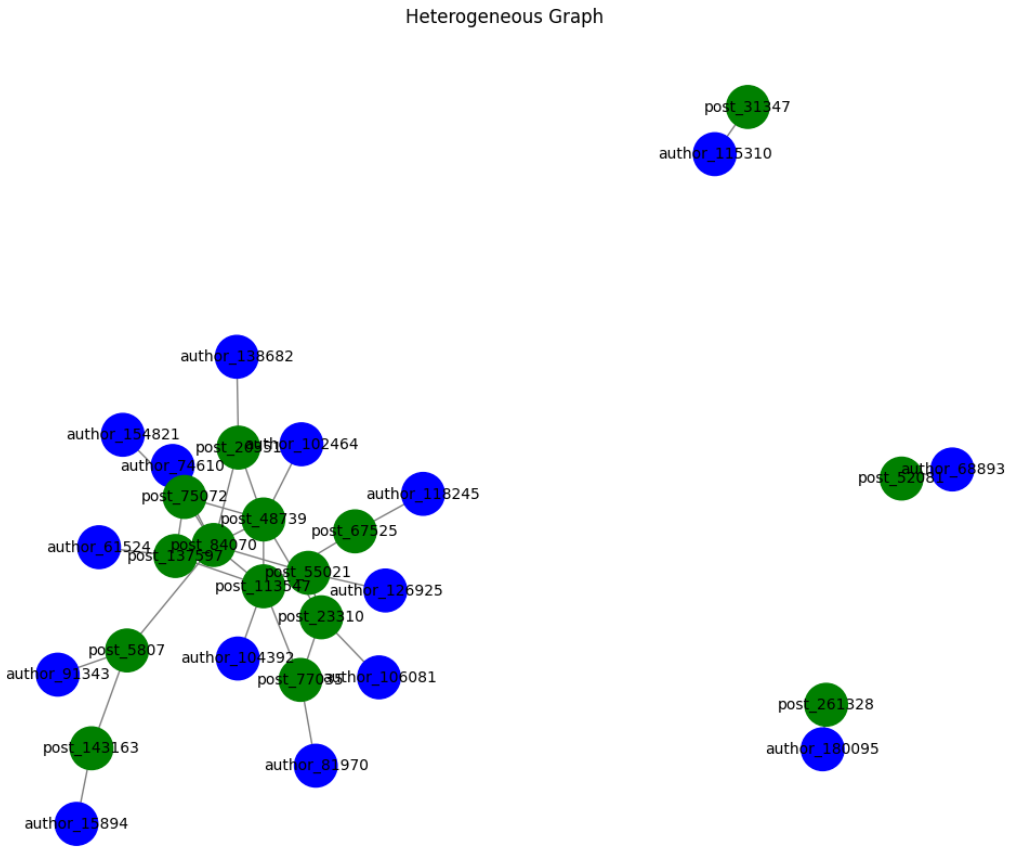

Figure 1: heterogenous graph representing the first 15 posts' nodes in the dataset and their edges.

Methodology

Overview of Chosen Architectures

Several models of GNNs were implemented with the purpose of making a comparison of those in the task of subreddit classification. The chosen models used in this work were GraphSAGE, GatedGNN, GCN, and GIN. Each adopts different graph convolutions as a way of processing information from the graphs. Specifically, GraphSAGE performs aggregation of the neighboring nodes' features by the functions of mean, sum, or pooling. This is scalable to large-size graphs. The GatedGNN architecture incorporates gated mechanisms, allowing it to dynamically select neighboring nodes to be aggregated based on learned parameters; hence, it has more flexibility in the aggregation process. On the other hand, GCN, a well-known framework, uses graph convolutions over node features informed by the graph adjacency matrix to capture information from the local neighborhood. Finally, the GIN model uses a more complex aggregation function, which makes it a powerful tool for learning node representations, especially since it is designed to discover subtle differences in node attributes.

Neural Network Architecture

All models incorporate two convolutional layers. Each convolutional layer is followed by a ReLU activation function and a final linear layer. In the forward pass, models are fed the embedded content of the post as the features of the node and only edges representing the similarity between posts are considered during training.

Implementation and Results

The models were implemented in PyTorch and trained on a dataset of subreddit classification. Each model was trained with similar settings, while some hyperparameters were changed in order to investigate their impact on performance. The major steps of training included the definition of the optimizer, loss function-cross-entropy loss-and the adjustment of parameters like learning rate and weight decay. Then, after training, all models were tested on a different test set. The metrics taken for evaluation include accuracy, precision, recall, F1-score, and cross-entropy loss; these metrics provided a comprehensive understanding of how well models could classify subreddit posts.

Hyperparameter Variations

We varied the hyperparameters in order to improve the models for the classification task and use the best configurations for further analysis. As such, different hyperparameters combinations were experimented with for each model to compare to the baseline performance such as:

- Number of output channels, e.g., 32, 128, or 512 for GCN.
- Aggregation functions (such as mean, sum, max for GraphSAGE).
- Secret layers and number of those (mostly for GatedGNN).
- Regularization parameters such as weight decay and learning rate.

Training and Testing Results

The models were trained for 500 rounds, and their performance was watched during the training and validation phases. Accordingly, the learning curves for training and validation losses and accuracies were drawn in order to see how the models learned and how they would work with new data. Sharing the final test results, including accuracy and other measures such as precision, recall, and F1-score, is carried out. The training can be summarized as follows:

- GraphSAGE: It was trained with an output channel size of 256 and showed a consistent increase in both training and validation accuracy.
- GatedGNN: With a hidden size of 400 and one layer it performed well, especially in understanding changing neighborhood connections.
- GCN: With an output channel size of 256, GCN showed stable performance in terms of classification accuracy and loss reduction.
- GIN: used a special type of neural network with a learning rate of 1e-4, which helped it to pick up detailed features resulting in strong

Model	Accuracy	Precision	Recall	F1 Score
GraphSAGE	0.9398	0.9399	0.9398	0.9398
GatedGNN	0.9313	0.9315	0.9313	0.9313
GCN	0.9396	0.9398	0.9396	0.9396
GIN	0.9302	0.9304	0.9302	0.9303

From the table, we can confidently say that the four graph neural network models—GraphSAGE, GatedGCN, GCN, and GIN—show very similar results on all evaluation metrics: accuracy, precision, recall, and F1 score. Overall, GraphSAGE is the best with an accuracy of 0.9398, precision of 0.9399, recall of 0.9398, and an F1 score 0.9398, showing it is the most effective model for this task. Next, GCN comes closely with precision and recall value of 0.9398 and 0.9396. GatedGCN and GIN follow with slightly worse performance: GatedGNN has an accuracy of 0.9313, while GIN has a lower accuracy of 0.9302. As their performance is remarkably similar, GCN and GraphSAGE are probably the best choices, with a slight advantage for GCN as it is generally simpler and less resource intensive. However, GraphSAGE might be a better should every ounce of performance is required.

Metrics and Learning Curves

The learning curves gave valuable information about how the models learn: GraphSAGE and GCN had the best early learning curves and converged quicker than GIN and GatedGNN. GatedGNN showed it could change its learning curve depending on how it grouped data from neighbors. The test metrics, including accuracy, precision, recall, and F1-score, showed that GCN and GraphSAGE performed better than the other models.

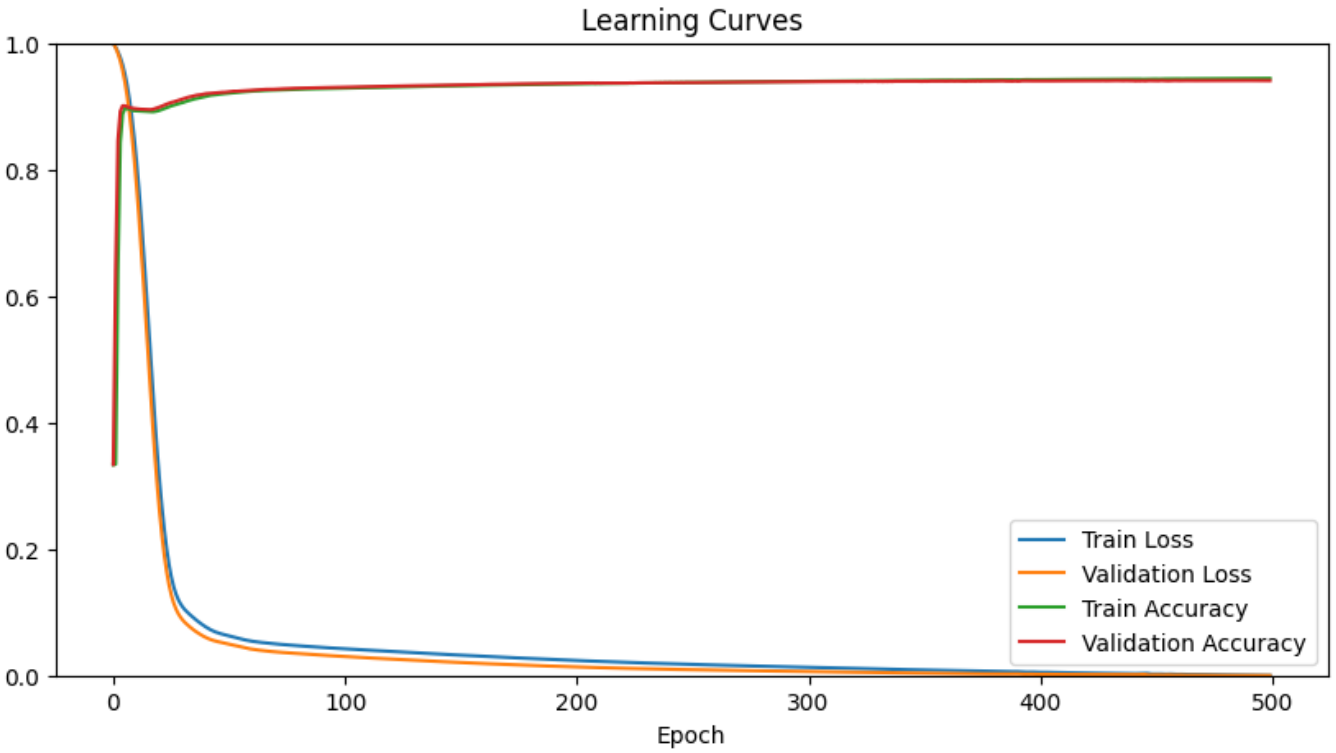

Figure 2: GraphSAGE training learning curves.

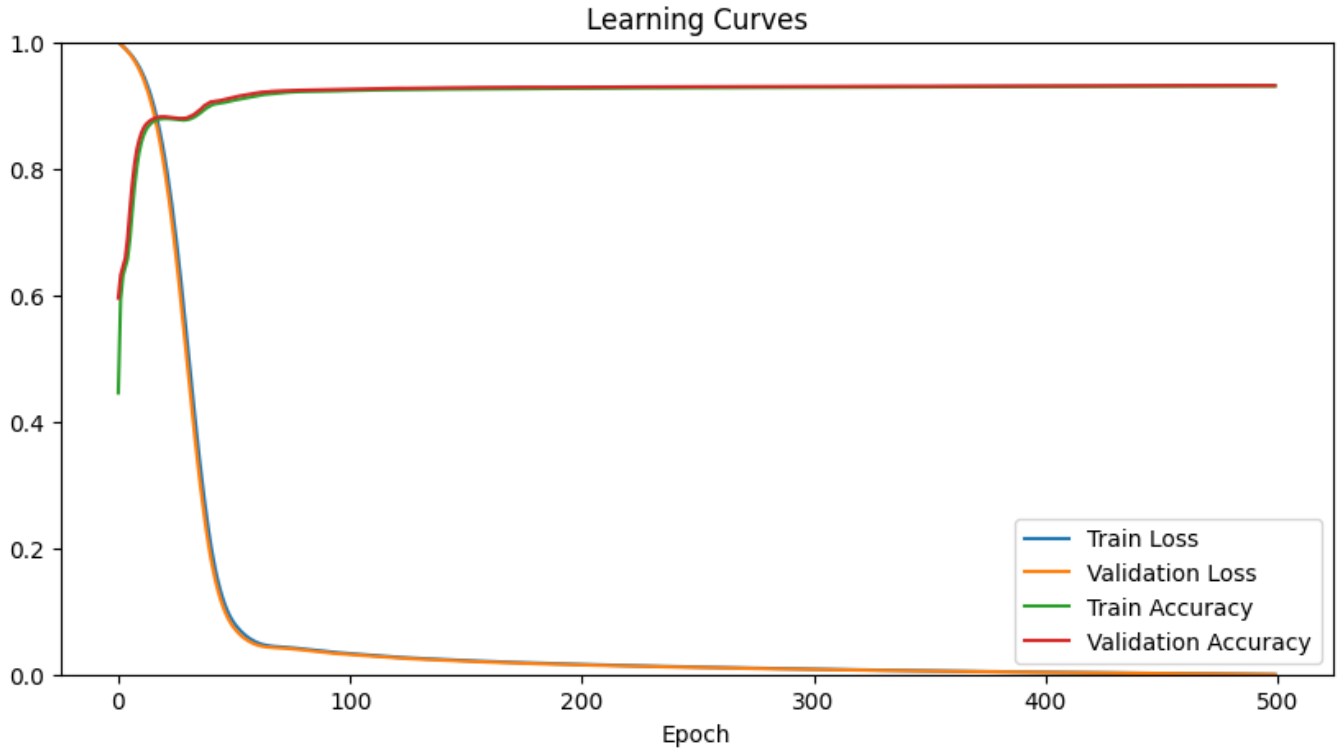

Figure 3: GGNN training learning curves.

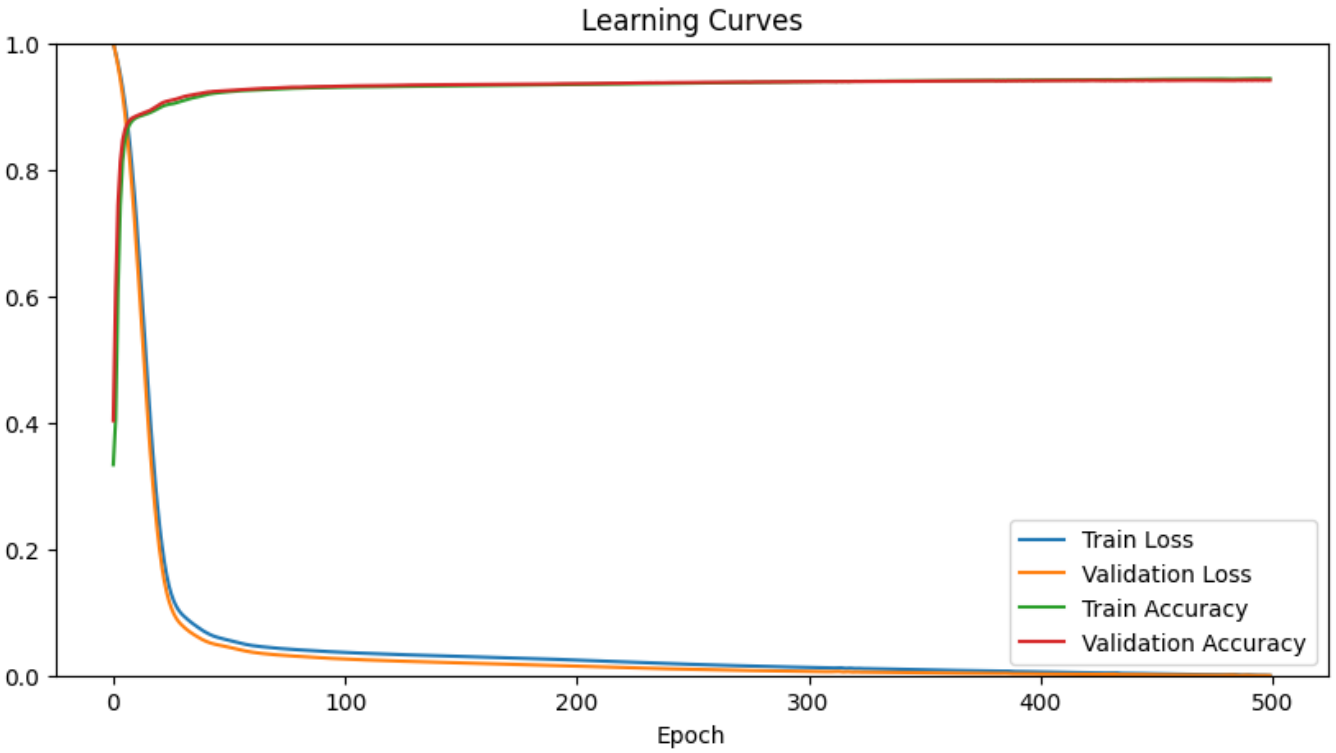

Figure 4: GCN training learning curves.

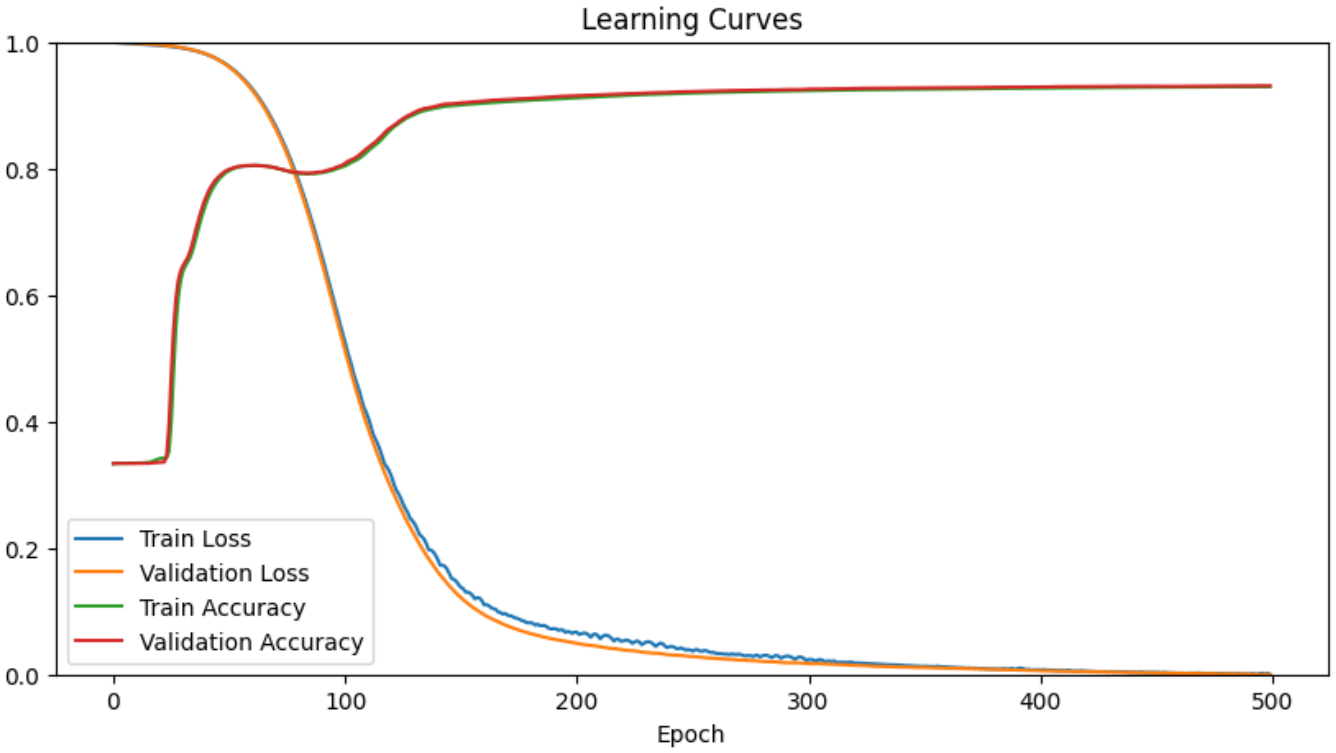

Figure 5: GIN training learning curves.

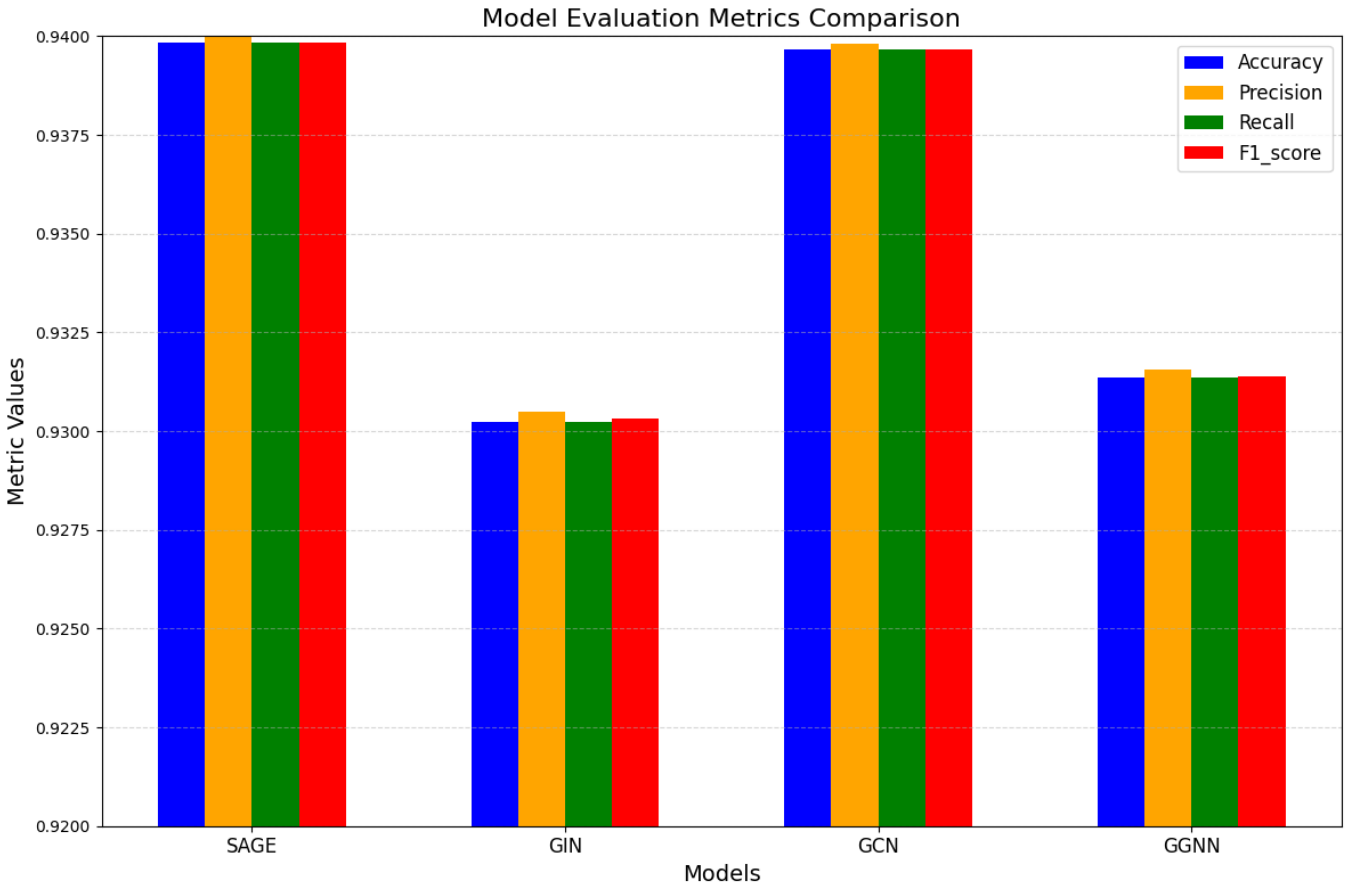

Figure 6: testing results against different evaluation metrics for each model.

Analysis

Computation Time and Space Consumption

We tracked memory consumption and measured the time each model takes to complete 500 epochs. The results show that GGNN by far takes the most time and memory to train. GIN comes in the second place, followed by GraphSAGE in the third, and GCN comes out as the most efficient architecture in terms of time taken for training and memory use. See figure 7 for detailed time and memory use measurements.

The significant resource utilization in GGNN is attributed to its selective retention and gating mechanism. GraphSAGE, on the other hand, tries to balance resource utilizations with effectiveness through the use of neighborhood sampling and node aggregation mechanism. GCN performs a single-pass aggregation and applies simpler linear transformations leading to more efficient resource utilization compared to the others. GIN performance is mostly influenced by the choice of the neural network utilized in the model.

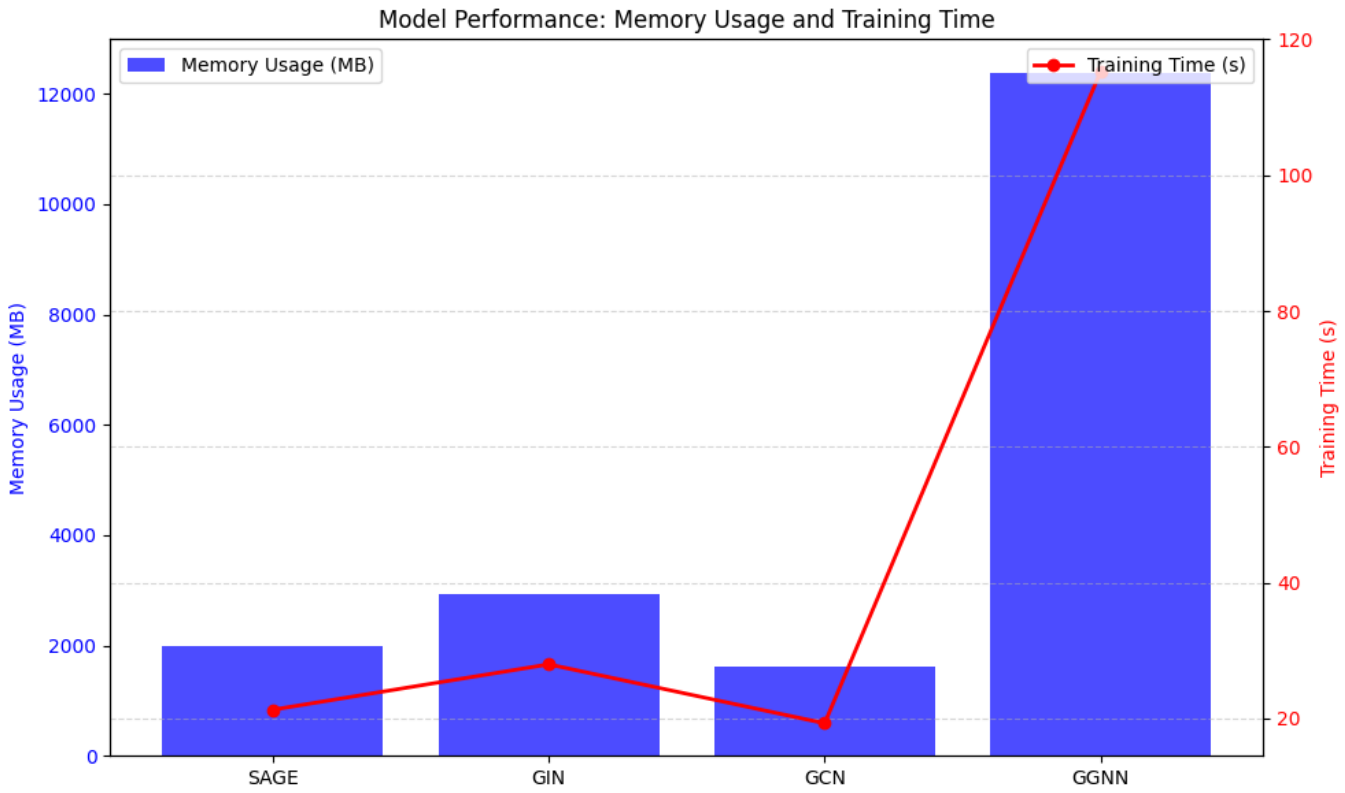

Figure 7: time and memory consumption utilized by each model during training.

Key Observations

The results showed that all models perform well in the task of classifying subreddit posts, but some, such as GCN and GraphSAGE, do so with greater accuracy. GraphSAGE can handle different types of graph structure because it is able to use different aggregation functions. GatedGNN and GIN perform notably worse especially in early epochs despite them being resource intensive in comparison to the other two.

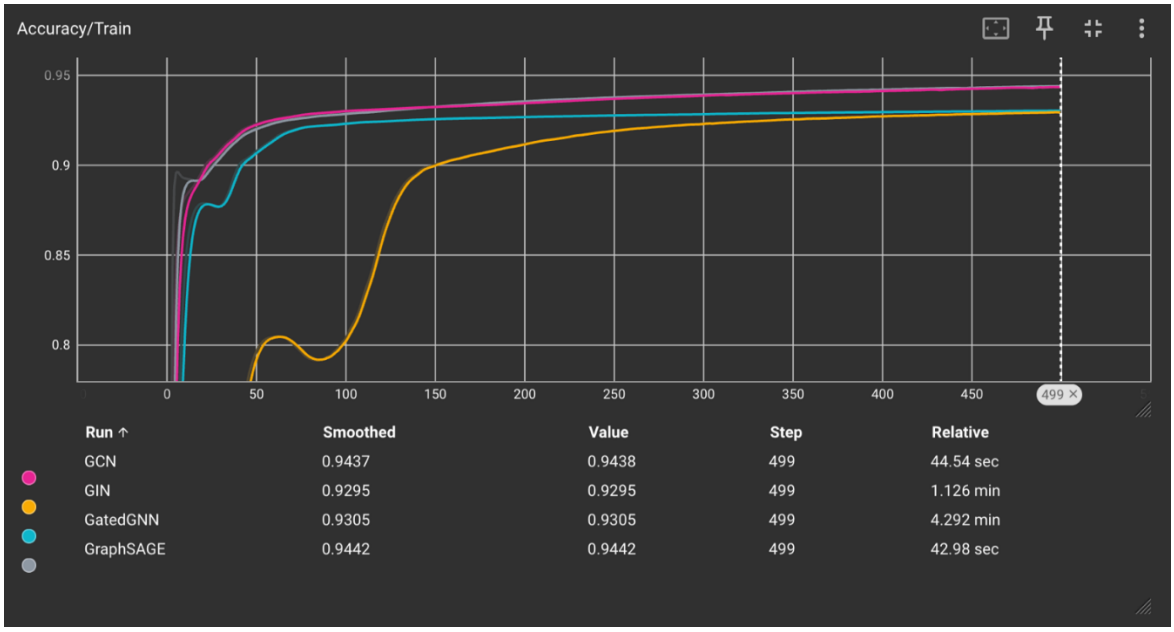

Figure 8: training accuracy for each model.

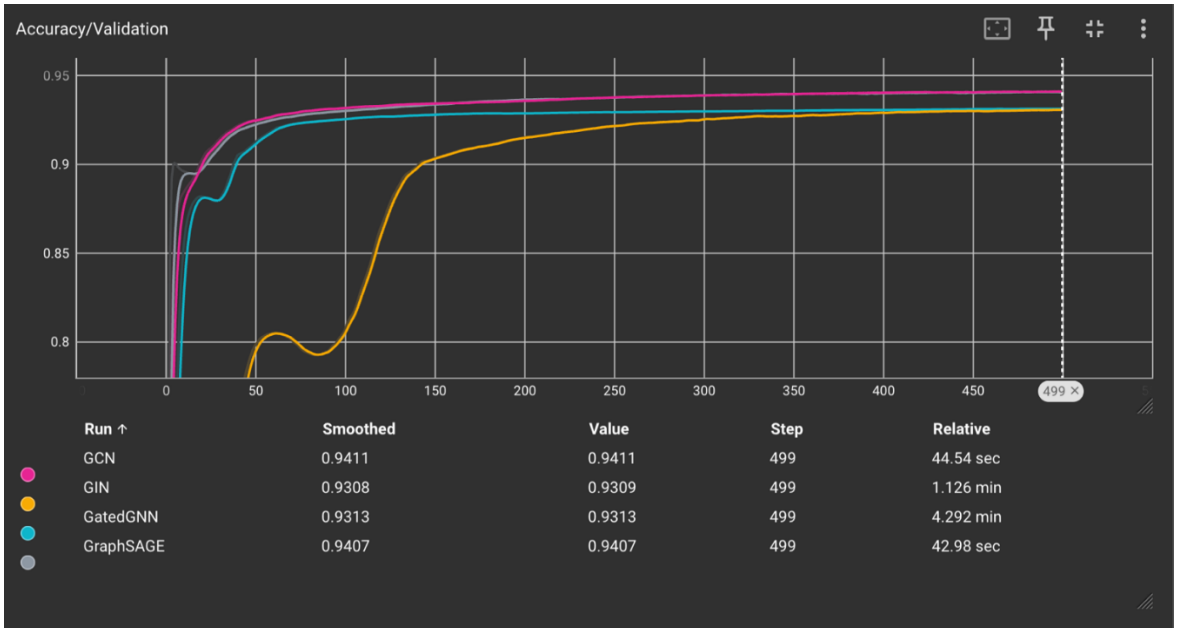

Figure 9: validation accuracy for each model

Evaluating Accuracy Across Different Hyperparameters

We evaluated each model against different numbers of hyperparameters and each of them did indeed influence the overall performance in some form or another. The following table shows the different hyperparameters evaluated as well as how they affected the performance on the validation set through 100 epochs of training:

Model	Hyperparameter Name	Tested values	Results	Comment
GraphSAGE (Amir224030)	Aggregation function	Max, sum	Max function yields noticeable better performance.	Unlike sum function, max function focuses on the most dominate and important feature contribution from neighbors making the model more robust and resilient to noise.
	Out channels (dimensionality of node embeddings produced by SAGE layer)	32, 512	Lower values for out channels produce stable and consistent performance across epochs. Higher values fluctuate the learning curve much more. Both yield similar performance with an average difference of about 0.1% to 0.2% across most epochs.	Higher values for the out channels introduce more opportunities for overfitting.
GatedGNN (Merihan226392)	Aggregation function	Max, sum	Max function, unlike sum, yields better performance and more consistent behavior across epochs.	Unlike sum function, max function focuses on the most dominate and important feature contribution from neighbors making the model more robust and resilient to noise.

	Number of layers	1, 2, 3	Lower number of layers had better performance and faster training time.	Higher number of layers the model more prone to overfitting, gradient vanishing or instability as well as potential over smoothing.
	Out channels (dimensionality of node embeddings produced by Gated layer)	420, 460, 500	Identical performance for 420 and 460 values and slight surge for 500.	The slight surge for 500 dimensionality value is hardly justifiable given that it requires significantly much more resources and time to train in comparison with lower values which are not way behind in terms of perormance.
GCN (Sameh218767)	Out channels (dimensionality of node embeddings produced by GCN layer)	32, 128, 512	Higher values for out channels tend to produce better performance compared to lower values, however, their performance abruptly fluctuate across epochs.	Higher values for out channels can learn and capture more complex pattern and relationships, however, they also make the model more complex introducing more chances of overfitting and sensitivity to noise which explains the fluctuation across epochs.
GIN (Omar228248)	Epsilon (ϵ)	0.1, 0.5, 1, 2	Different values of epsilon do not seem to affect the performance much and the generally converge to remarkably similar final performance by the end of training. However, higher values tend to show slight fluctuation at the early and last epochs.	Higher values of epsilon amplify the gradients associated with each node's features; hence, this can lead to more significant weight updates.

Confusion Matrix

In addition to learning curves, confusion matrices were created for each model to check how well they classified. These matrices were visualized using seaborn's heatmap function, which gave a clear view of true labels compared to predicted labels and helped find areas where the models did well or had problems. All models tend to confuse many instances of class 0 with class 2 which is an indicator for possible room for improvement for the data itself in terms of preprocessing.

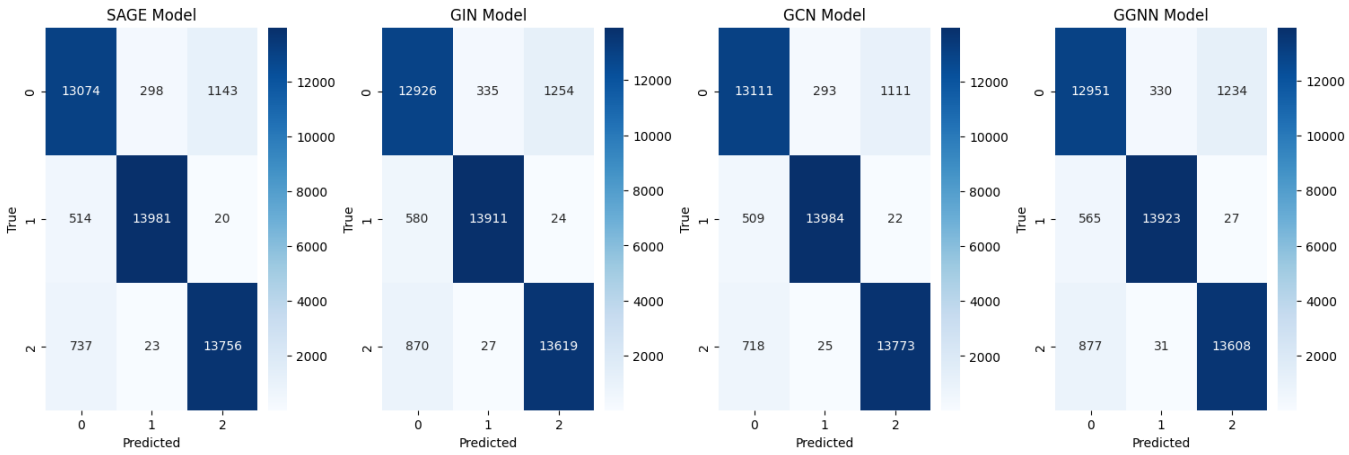

Figure 10: confusion matrix comparing the performance of each model.

Conclusion

This work demonstrates how well different GNN models work in the classification task for subreddits and how similar their results can be. In general, GraphSAGE and GCN were best in terms of accuracy and adaptability. Hyperparameter adjustment proved important in making the models better, and tools such as TensorBoard and confusion matrices aided in understanding the model behavior. We conclude the GCN strikes the best balance between performance and resources use efficiency, followed by GraphSAGE, GIN, and GatedGNN.