Yash Avinash Patole

CWID: 10460520

CS-541 Artificial Intelligence

Mini Project

*Title:*

Environment Setup

*Under the Guidance of:*

Prof. Abdul Rafae Khan

*Stevens Institute of Technology*

**Title :** The Logical Expressiveness of Graph Neural Networks.

Authors: Pablo Barcelo´ (IMC, PUC & IMFD Chile), Egor V. Kostylev (University of Oxford), Mikael Monet ¨ (IMFD Chile), Jorge Perez ´ DCC, UChile & IMFD Chile Juan Reutter DCC, PUC & IMFD Chile Juan-Pablo Silva DCC, UChile

Name: Yash A. Patole

StevensId: [ypatole@stevens.edu](mailto:ypatole@stevens.edu)

CourseName: CS 541 Artificial Intelligence

**Abstract:** In this paper it states that the capacity of graph neural networks (GNNs) to differentiate nodes in graphs has recently been measured using the Weisfeiler-Lehman (WL) graph isomorphism test. However, this definition does not resolve the question of which Boolean node classifiers may be described by GNNs. In this paper it tackles the problem by using various formulas such as the logic of FOC2, a well-studied first-order logic fragment. FOC2 is linked to the WL test and, as a result, to GNNs. They have focused on explaining the class of GNNs where this class shows and updates the features of each node in the graph. But here they show that this class of GNNs is too weak to capture all FOC2 classifiers, and we give a syntactic description of the greatest subclass of FOC2 classifiers that AC-GNNs can capture. This subclass corresponds to a widely used logic in the knowledge representation community. The paper then shows demonstrate on synthetic data complying to FOC2 formulae, AC-GNNs fail to fit the training data, whereas ACR-GNNs can generalize even to graphs of sizes not encountered during training.

This means that ACGNNs aren't powerful enough to capture all FOC2 formulations. Then we look for what needs to be added to AC-GNNs so that all FOC2 can be captured. I have demonstrated that adding readout layers is sufficient for updating node features not just in terms of its neighbors, but also in terms of a global attribute vector. ACR-GNNs are GNNs with readouts. The final results   experiments propose that support the findings of AC-GNNs which fail to fit in on synthetic data conforming to FOC2 but not to graded modal logic, but ACR-GNNs can generalize to graphs of sizes not observed during training.

The following is the main consideration of the paper:

1. We describe in detail the FOC2 formula fragment that can be expressed as ACGNNs. This section belongs to graded modal logic (de Rijke, 2000), or, to put it another way, the ALCQ description logic, which has sparked a lot of attention in the field of knowledge representation (Baader et al., 2003; Baader & Lutz, 2007).
2. Next, we extend the AC-GNN architecture by allowing global readouts, in which each layer computes a feature vector for the entire graph and combines it with local aggregations; these aggregate-combine-readout GNNs are called aggregate-combine-readout GNNs (ACR-GNNs). This proposal is for relational reasoning over graph representations is a subset of these networks. We show that an ACR-GNN can capture each FOC2 in this scenario.

GitHub link of the research paper: <https://github.com/juanpablos/GNN-logic>

**Introduction:**

Graph neural networks are a sort of neural network that has recently acquired popularity for a wide range of structured data applications, such as molecular categorization, knowledge graph completion, and Web page ranking. The authors' major goal in writing this study is to experiment with and comprehend the connections between neurons that aren't random but nevertheless have an impact on the structure of the incoming data.

The authors of these papers independently observe that the node labeling produced by the WL test always refines the labeling produced by any GNN. More precisely, if two nodes are labeled the same by the algorithm underlying the WL test, then the feature vectors of these nodes produced by any AC-GNN will always be the same. Answering this question, we recommend focusing on logical classifiers—that is, unary formulae expressible in first order predicate logic (FO): such a formula categorizes each node v based on whether the formula holds for v or not. This focus allows us to connect GNNs to declarative and well-understood formalisms, as well as derive inferences about GNNs based on a large body of logic research. For instance, if two GNN architectures are captured by two logics, all knowledge about the links between those logics, such as equivalence or incomparability of expressiveness, can be transferred to the GNN setting instantaneously.

The expressivity of graph neural networks is discussed in depth in this paper (GNNs). The authors show that the expressivity (aggregate and combine) of AC-GNNs can only express logical classifiers that can be articulated in graded modal logic. ACR-GNNs (aggregate, combine, and readout) can capture FOC2 (logical classifiers expressed with two variables and counting quantifiers) by adding readouts. The second theorem leaves open the possibility of ACR-GNNs capturing logical classifiers other than FOC2.

The AC-GNN architecture is simplified by enabling global readouts, in which each layer computes a feature vector for the entire graph and combines it with local aggregations; these aggregate-combine-readout GNNs, or ACR-GNNs, are called aggregate-combine-readout GNNs. These networks are a variant of Battaglia et al's networks for relational reasoning over graph representations. In this context, the paper show that an ACR-GNN can capture each FOC2 formula also show that an ACR-GNN can capture each FOC2 formula with a single readout.

Finally, in the experiment I demonstrate that when we learn from instances, the theoretical expressiveness of ACR-GNNs, as well as the differences between AC-GNNs and ACR-GNNs, can be noticed. The results that AC-GNNs fail to match the training data on synthetic graph data adhering to FOC2 formulae, whereas ACR-GNNs can generalize to graphs of sizes not observed during training.

**Data:**

The graphs used in the paper are in the zip-file datasets.zip. Just unzip them to src/data/datasets.

The script expects 3 folders inside src/data/datasets named p1, p2 and p3. These folders contain the datasets for the properties [](https://camo.githubusercontent.com/8a8324063b17c84ab16bedd4e762b5a5ee4597c014cb484ef1d3419709568381/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c616c7068615f31), [](https://camo.githubusercontent.com/0ebc84e6bfcf55ea549b422275e543352d361f07b1464fd812fd890bd1dc2b94/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c616c7068615f32) and [](https://camo.githubusercontent.com/97651653395f55e636a658477de5c1630678a458c5f157689d4a87b1c5072a18/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c616c7068615f33) described in the appendix F on data for the experiment with classifier [](https://camo.githubusercontent.com/d5541ef6dc6224f13753cb306b17102598031ec4967cea76ba4d7df13960e164/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c616c7068615f69287829) in equation.

In the code there these datasets are accepted parsed and there is a small description of the arguments in generate\_dataset.

While there are .txt in file in each p1, p2 and p3 which has test and train data respectively. Size of the data are as follows:

P1: 1) size of train data is 230087. 2) size of test data is 51377.

P2: 1) size of train data is 230232. 2) size of test data is 51198.

P3: 1) size of train data is 229788. 2) size of test data is 51239.

Link to the dataset: <https://github.com/juanpablos/GNN-logic>

We can find the link here and also I will attach the zip file of the dataset with this submission.

**Tools & Technologies**: The details of the software/library packages tqdm==4.35.0

scipy==1.3.1

numpy==1.17.1

scikit-learn==0.21.3

matplotlib==3.1.1

networkx==2.3

torch==1.2.0

torch-cluster==1.4.4

torch-scatter==1.3.1

torch-sparse==0.4.0

torch-spline-conv==1.1.0

torch-geometric==1.3.1

requests==2.22.0

For this project we will use Google Colab and it’s technologies.

Following links will be required to save:

1. /content/drive/MyDrive/GNN-logic/src/logging/results/p1-0-0-acgnn-aggA-readA-combT-cl0-L1.log
2. /content/drive/MyDrive/GNN-logic/src/logging/results/p1-0-0-acgnn-aggA-readA-combT-cl0-L1.log.train
3. /content/drive/MyDrive/GNN-logic/src/logging/results/p1-0-0-acgnn-aggA-readA-combT-cl0-L1.log.test

Hardware Requirements: This project requires GPU (cuda and other torch packages). My GPU is Nvidia RTX3080 and processor is AMD Ryzen 5000.

**Data:**

|  |  |  |
| --- | --- | --- |
| **P1** | **230087** | **51377** |
| **P2** | **230232** | **51198** |
| **P3** | **229788** | **51239** |

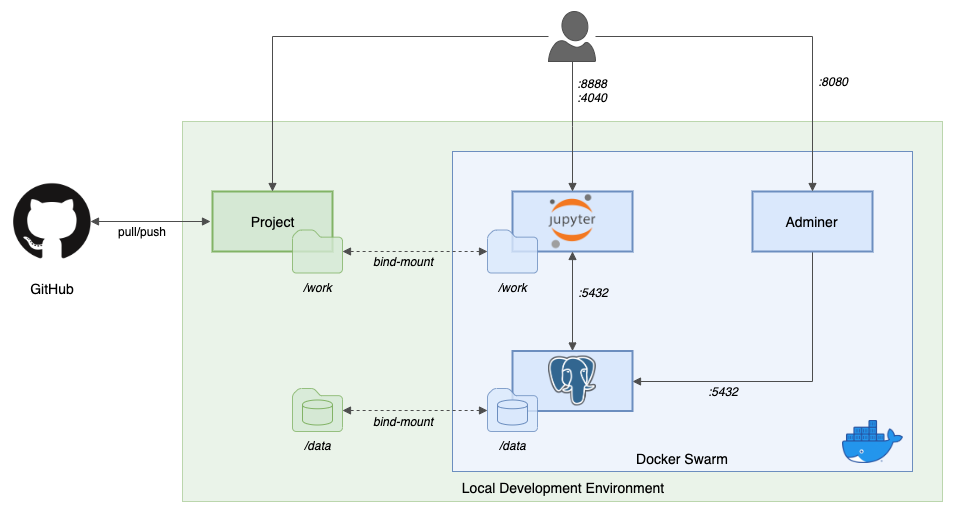
For data Pre-Processing this project uses two files named [argparser\_real\_data.py](https://github.com/juanpablos/GNN-logic/blob/master/src/datasets/argparser_real_data.py) and datasets.py to parse and get the inputs ready for processing epochs.

**Experiments:** The code helps to get logical expressiveness of graph neural networks. In this code it considers the simple FOC2 formula: (x):= Red(x) y Every red node in a graph must satisfy the condition Blue(y) if the graph contains at least one blue node. The code first considers half of the graphs in each set (training and testing) containing no blue nodes, and 50% containing at least one blue node (about 20% of nodes are blue every set has a true class). As my initial test I will provide following training data outcomes which explains the learning rate, epochs of the our training dataset.

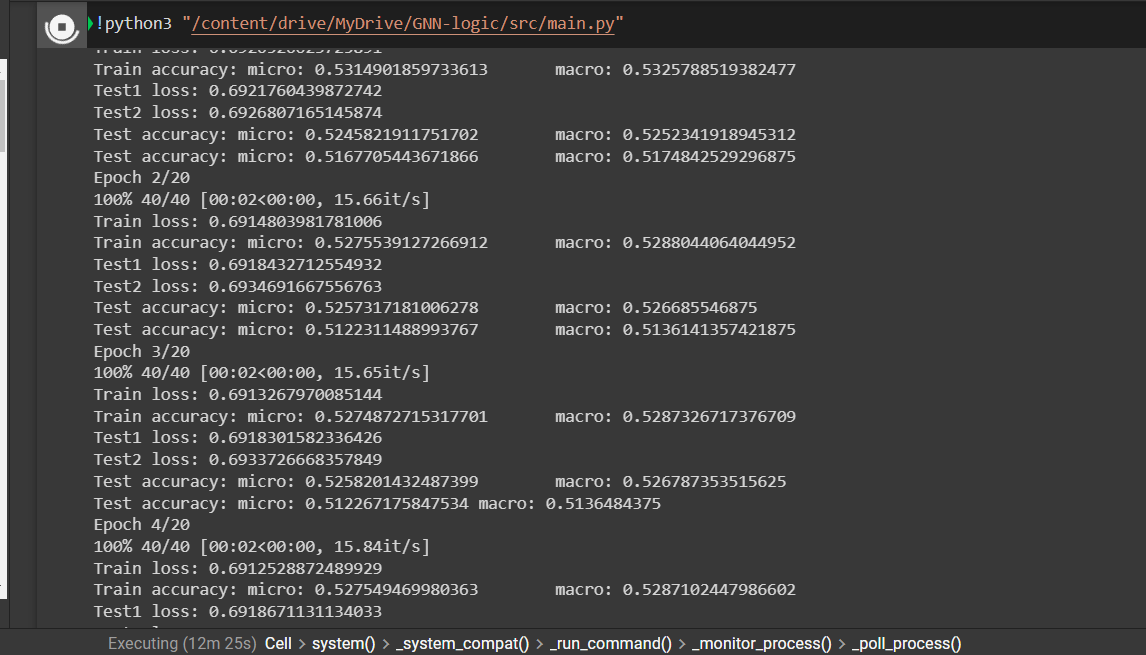
train\_loss,test1\_loss,test2\_loss,train\_micro,train\_macro,test1\_micro,test1\_macro,test2\_micro,test2\_macro

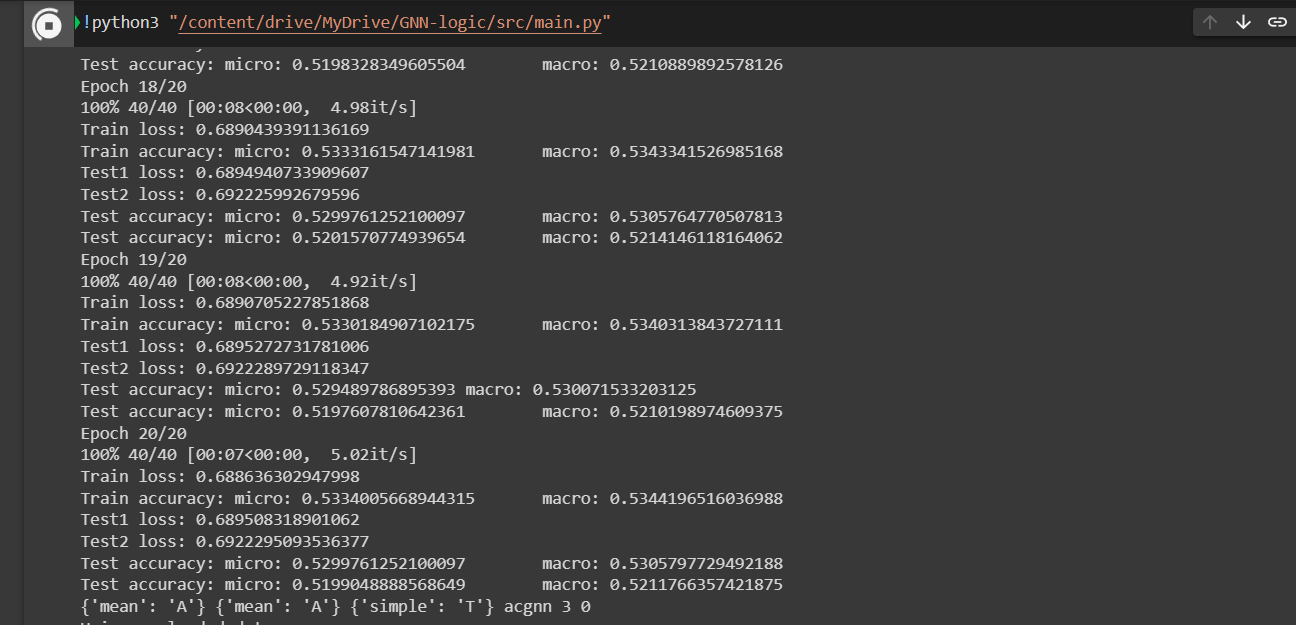
0.6901471019, 0.6904880404, 0.6916720867, 0.53859858, 0.53905009, 0.53607746, 0.53637073, 0.52487661, 0.52607605

Following is similar network architecture that will be required where instead of postgres in this project I am using google colab and jupyter lab to get this project onboarded and modules are installed respectively.



Results: Following are preliminary results of this project that I managed to execute on my computer successfully. Following outputs contains 20 epochs for dataset P1. More testing, including datasets P2 and P3 are needed to be done.





**Experiments & Results**

**Method:**

The method used in this paper is encodings node colors in the graph, from a finite set of colors. While using neighborhood NG(v) of a node where v ∈ V is the set {u | {v, u} ∈ E}. GNNs can be classified using a variety of aggregate, combination, and classification functions (Hamilton et al., 2017; Kipf & Welling, 2017; Morris et al., 2019; Xu et al., 2019). Consider the sum of the feature vectors as the aggregation, which is a basic yet frequent choice.

function, as well as a combined function

f = (x1, x2)

x1C

x2A(i) + b = I + x2A(i)

(2)  where C stands for (i) as well as (i) are parameter matrices, If is a non-linearity function, such as relu or sigmoid, and is a bias vector. Using these functions, we call it an AC-GNN. In addition, if all AGG are present in an AC-GNN, it is said to be homogenous (i) all identical, and they're all COM (i) are the same (common) the same settings are used in each tier.

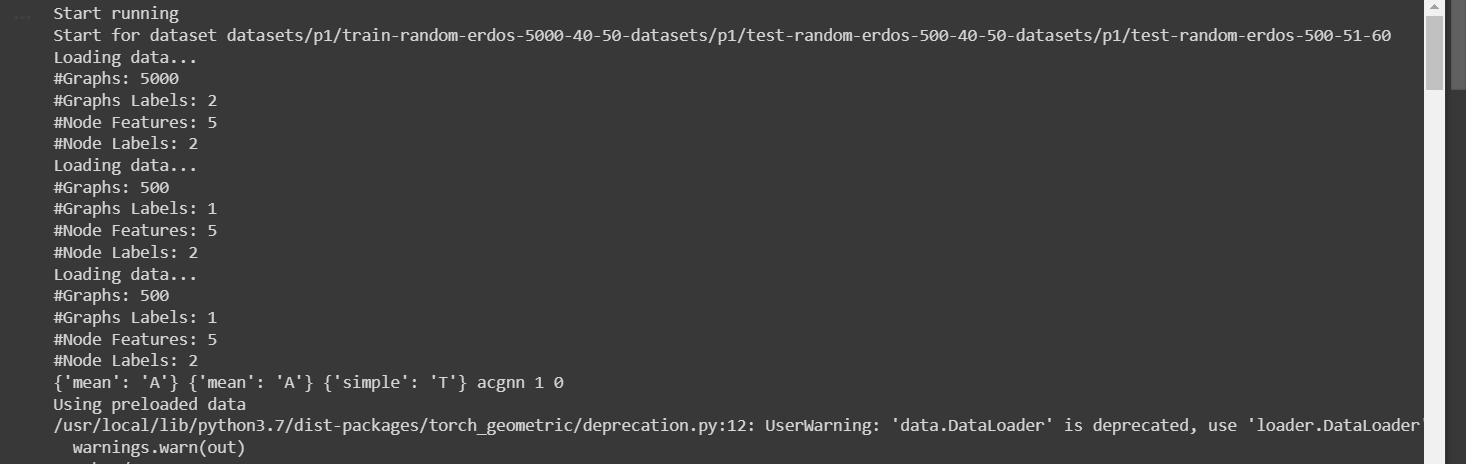
This formula has two quantified variables, y and z, and one free variable, x, that is not constrained by any quantifier of the form or. Formulas with one free variable are evaluated over the nodes of a network in general. For example, in nodes v with the hue Red with both a Blue and a Green neighbor, the above algorithm evaluates to true exactly. In this scenario, we state that G's node v satisfies and mark this with (G, v) |=α.

The proof of Theorem 5.1 mentioned in the paper creates GNNs whose number of layers is determined by the formula being captured—readout functions are utilized an infinite number of times in ACR-GNNs to capture various FOC2 classifiers. Given how expensive a global calculation can be, one can wonder if it's truly necessary, or if the complexity of such classifiers can be handled with only a few readouts.

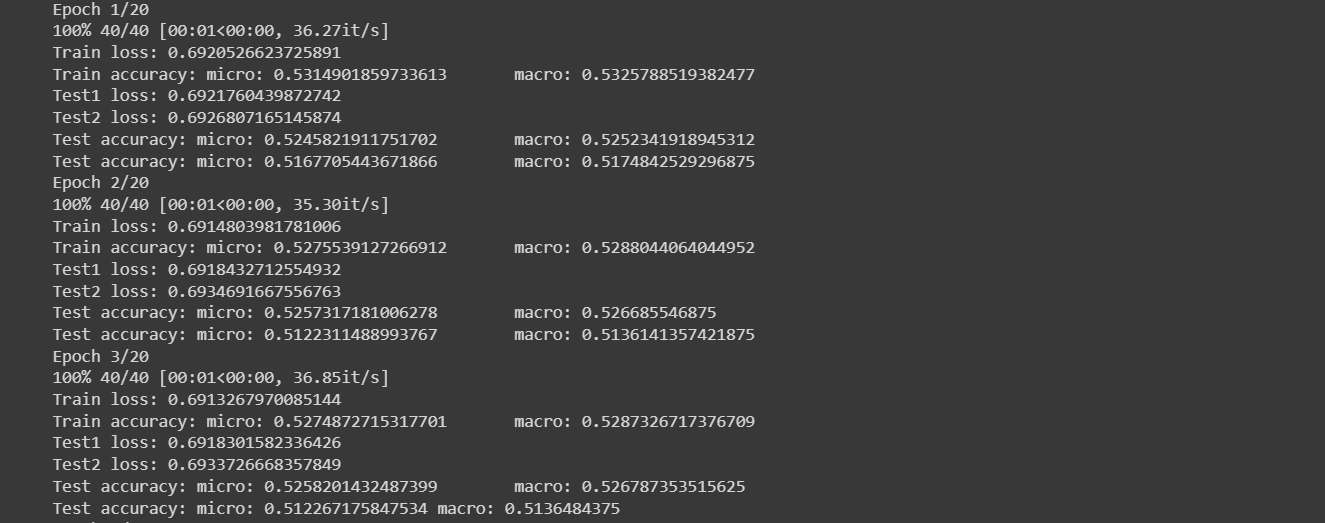
**Experiments:**

Considering the following simple FOC2 formula: (x):= Red(x) y Every red node in a graph must satisfy the condition Blue(y) if the graph contains at least one blue node. We used line-shaped graphs and Erdos-Renyi (E-R) random graphs with various connectivity to test our hypothesis. In each set (train and test), we consider 50 percent of graphs with no blue nodes and 50 percent with at least one blue node (about 20% of nodes in each set are in the true class). Single-layer ACR-GNNs have already demonstrated excellent performance for both types of graphs (ACR-1 in Table 1). Given the simplicity of the property being examined, this was to be anticipated. AC-GNNs and GINs, on the other hand (represented in Table 1 as AC-L and GINL, respectively, indicating AC-GNNs and GINs) also, GINs with L layers) have a hard time fitting the data. In the instance of the even with 7 layers, they were unable to fit the train data into a line-shaped graph. In this scenario, the performance with 7 layers was significantly better than that of random graphs. A closer examination of the experiment interprets discovered that AC-GNNs performed better for varied connectivity of E-R graphs when we use more dense graphs to educate them.

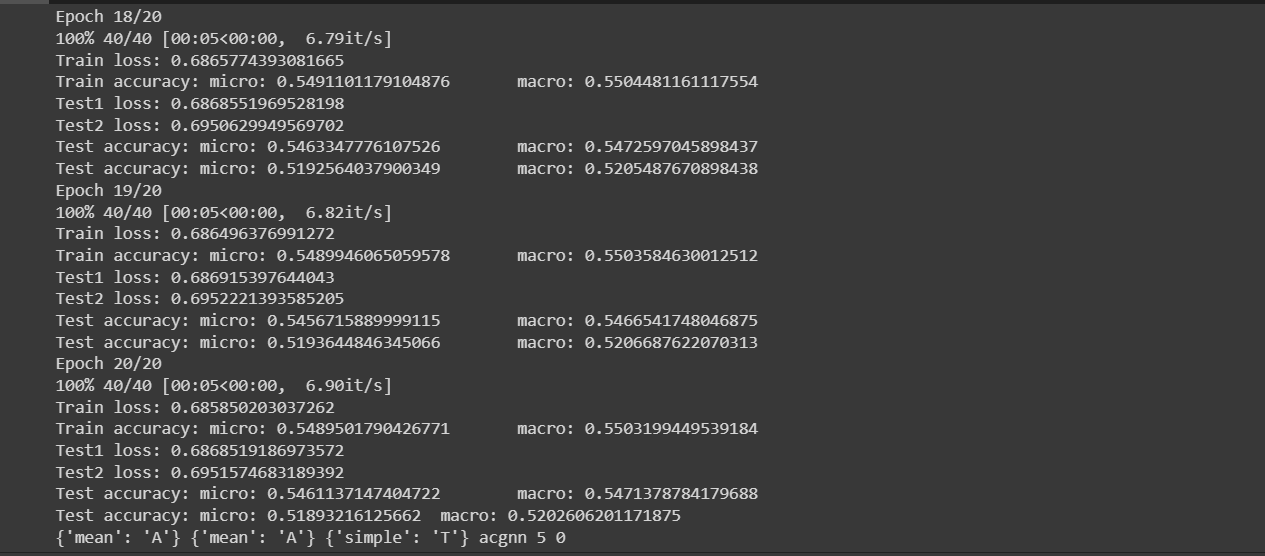
Following experiments were conducted in three different datasets provided in the repository.



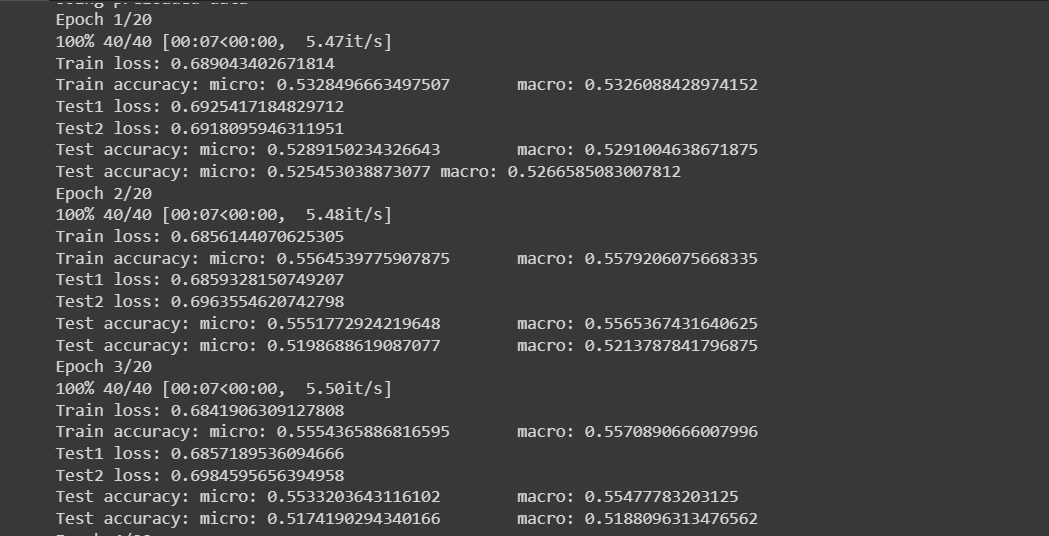
Epoch for P1 dataset:



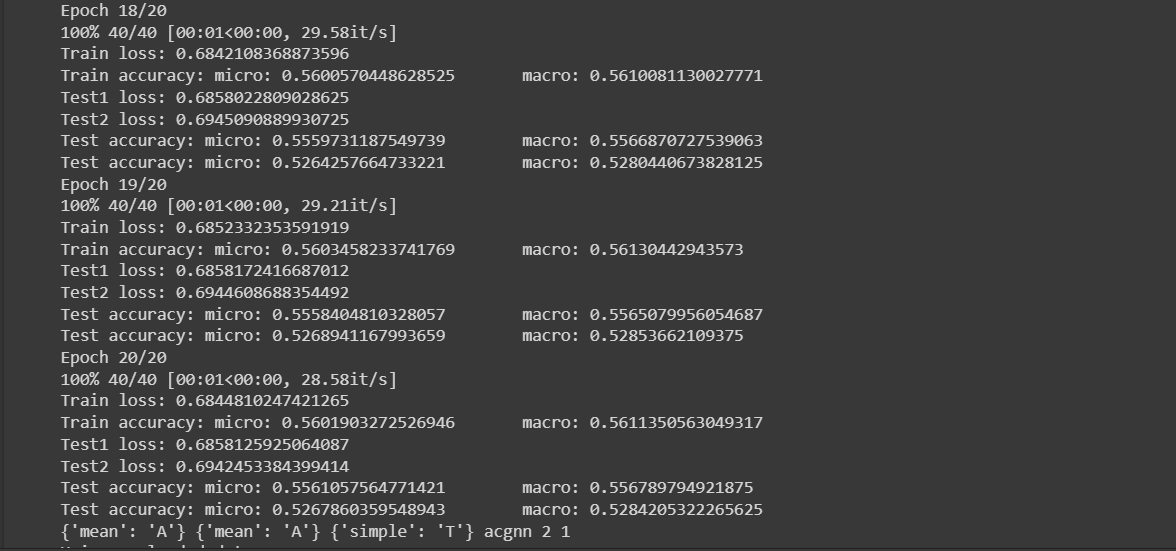
Final output after 20 iterations:



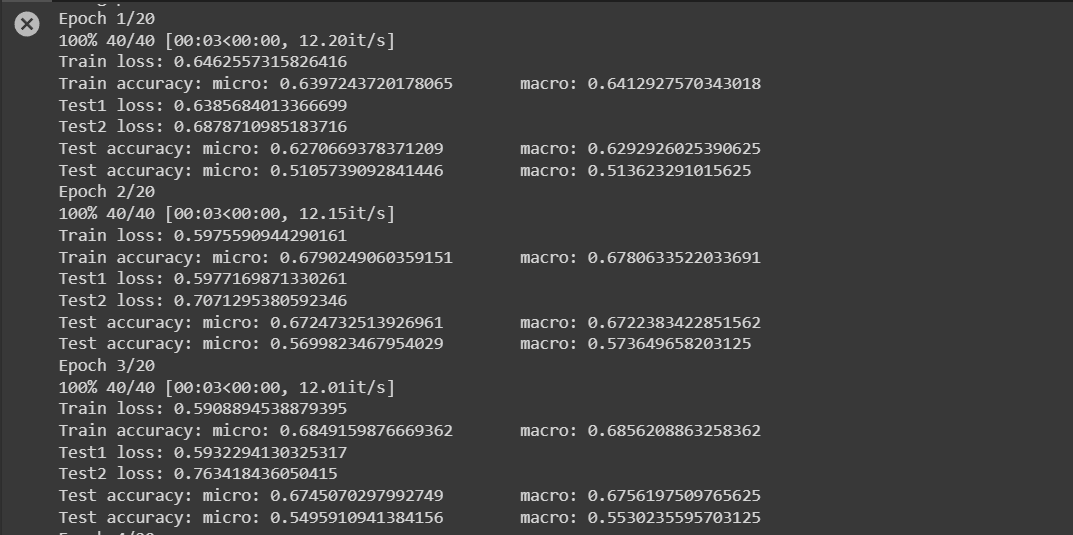
Epoch for P2 dataset:



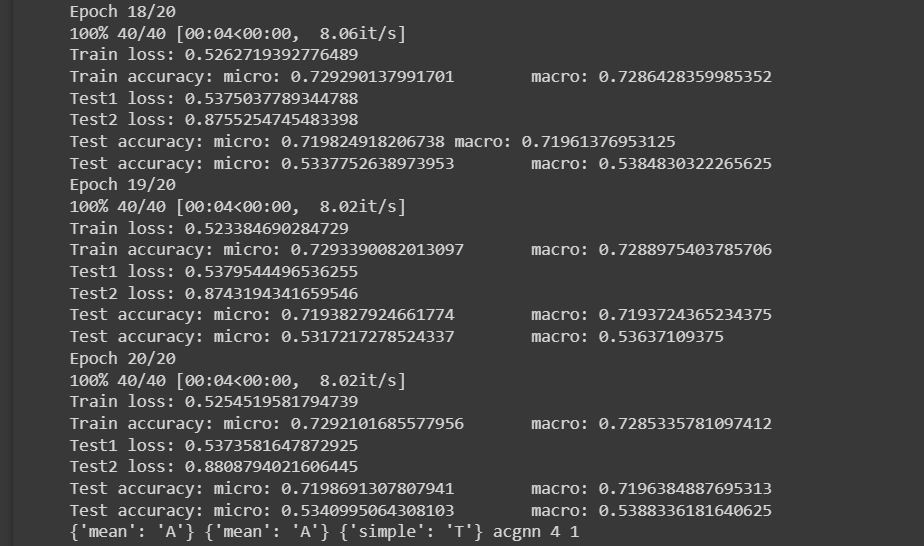
Final output of P2 dataset after 20 iterations:



Epoch for P3 dataset:



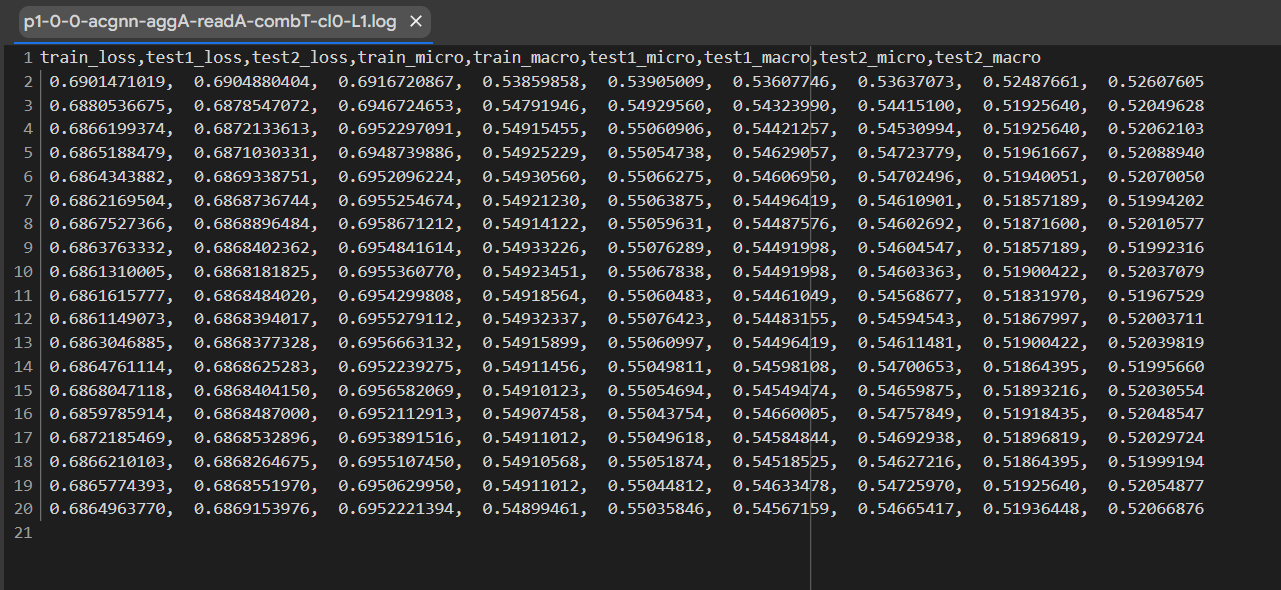
Final output of P3 dataset after 20 iterations:



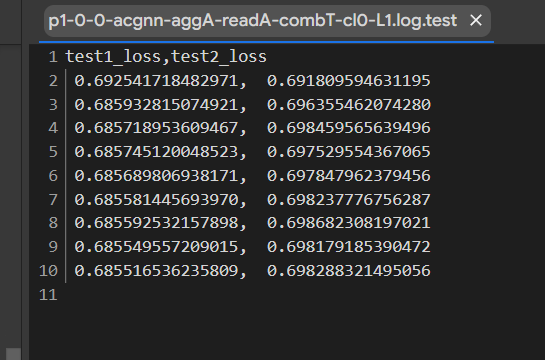
**Results:**

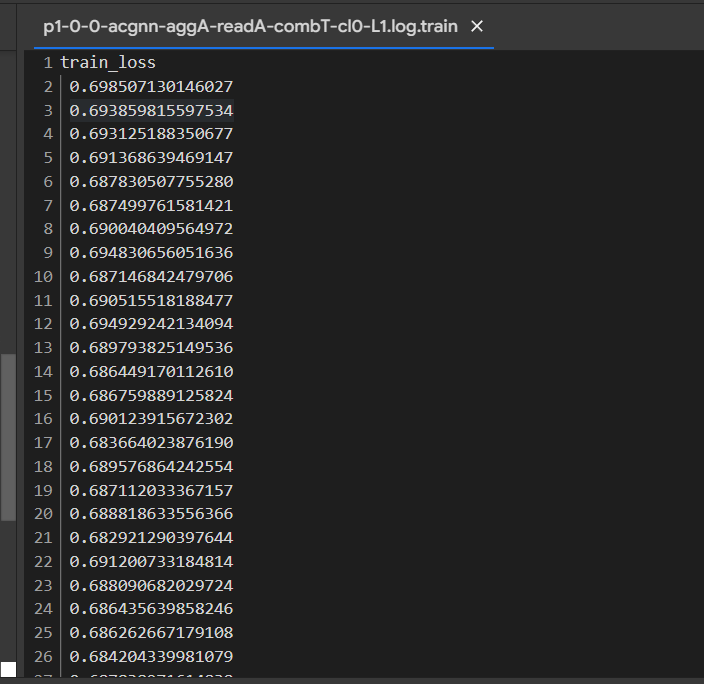
After training all the datasets final result files are saved in the logs and which can be viewed as following images.

Log file for training all the graph coordinates present in P1, P2 and P3 datasets which can give a perfect overview of local aggregations to distances up number of layers. This combined with the fact that random graphs that are denser make the maximum distances between nodes shorter, may explain the boost in performance for AC-GNNs.

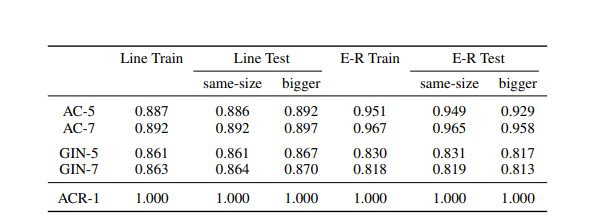


Loss in test1 and test 2.

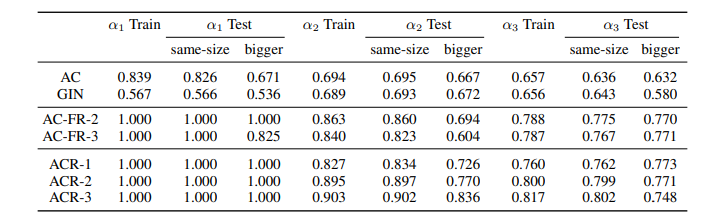




Results on synthetic data for nodes labeled by classifier α(x) := Red(x) ∧ ∃y Blue(y) are obtained by above files and can be seen as the table below:



Similarly, as per the equation (6) in the paper Results on E-R synthetic data for nodes labeled by classifiers αi(x) can be seen in following table.



**Problems/Issues Faced:** I faced few problems while setting this up on my local machine. First of all I tried to get this on VScode which did not work due to some version error. Then I decided to get this project on Jupyter and google colab. This step helped me setting up my environment successfully. While installing few libraries like torch-cluster, torch-scatter, scipy(numpy) I faced issues due to some errors. One of which included cannot get wheel setup for torch. Main issue I faced was creating a log file in my system since they had designed it for their own systems. I had to change the code according to my system to save the logging result to my system. Every time I ran the code it failed to detect my GPU for some reason, I tried to resolve this using it on google colab. Mainly torch version used in the code and given in requirement.txt is quite old and creates many issues every time when I start training the model. All the changes had to be made every time whenever I trained the model because colab does not save the modules installed in prior testing.

**Conclusion:**

The main contribution of this paper is the identification of logical classifiers that can be represented within an AC-GNN, a fragment of first order logic known as graded modal logic, as well as an extension of AC-GNNs that can capture a strictly more expressive fragment of first order logic known as ACR-GNN. Both of these conclusions are backed up by formal proofs and derivations, giving the study not only theoretical importance but also intuitive insights into the reasons for the AC-GNN limits. Furthermore, with the exception of a few datasets where no significant difference in performance between AC-GNNs and ACR-GNNs was discovered, the reported studies indicate the practical relevance of the findings. When classifying nodes in a graph, the results demonstrate the theoretical benefits of combining local and global information. Recently, Deng et al. exploited global-context aware local descriptors to categorize objects in 3D point clouds, demonstrating these benefits in practice.

On other hand, researching the theoretical features of GNNs makes them more transparent and illustrates the wide range of applications they can be applied to. On the other hand, while the author's GNN variant is a special case of an existing class of GNNs, the motivation for its creation is distinct and follows organically from the paper's discussion.

Graphs in k-GNNs are structures that connect k-tuples of nodes rather than merely pairs. We intend to investigate how our logical classifier findings relate to k-GNNs, particularly in relation to the logic FOCk, which extends FOC2 by allowing formulae with k variables for each fixed k > 1.

This study, I believe, makes a substantial contribution.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*All of the above experiments can be seen and viewed in the following links:*

*Link for the entire repository:*

[*https://github.com/yash171298/CS-541-Artificial-Intelligence/tree/main/GNN-logic-experiment-project*](https://github.com/yash171298/CS-541-Artificial-Intelligence/tree/main/GNN-logic-experiment-project)

*Link for log file having results saved after training the model:*

[*https://github.com/yash171298/CS-541-Artificial-Intelligence/tree/main/GNN-logic-experiment-project/src/logging/results*](https://github.com/yash171298/CS-541-Artificial-Intelligence/tree/main/GNN-logic-experiment-project/src/logging/results)

*Link for experiments performed before having the final output:*

[*https://github.com/yash171298/CS-541-Artificial-Intelligence/blob/main/GNN-logic-experiment-project/final\_experiment\_GNN.ipynb*](https://github.com/yash171298/CS-541-Artificial-Intelligence/blob/main/GNN-logic-experiment-project/final_experiment_GNN.ipynb)