Author Collaboration Prediction for OGBL-COLLAB

AI60007: Graph Machine Learning

Term Project Report – Group 03

Gajula Sai Chaitanya	Bale Veeresh Siva Sai	Eate Sohil	Yegurla Sumanth
(18CS30018)	(18AE30001)	(18AE30022)	(18AE30028)

Problem Statement:

The ogbl-collab dataset is an undirected graph, representing a subset of the collaboration network between authors indexed by MAG. Each node represents an author and edges indicate the collaboration between authors. The task is to predict the future author collaboration relationships given the past collaborations. The goal is to rank true collaborations higher than false collaborations.

Importance of the problem statement:

The problem statement may be interesting particularly for the researchers who may be interested to know future collaborations among various authors which may be useful to analyse successful collaborations. For example, consider a scenario of a research institute, knowing probable collaborations between two authors helps to decide whether to fund a particular research facility.

Dataset splitting:

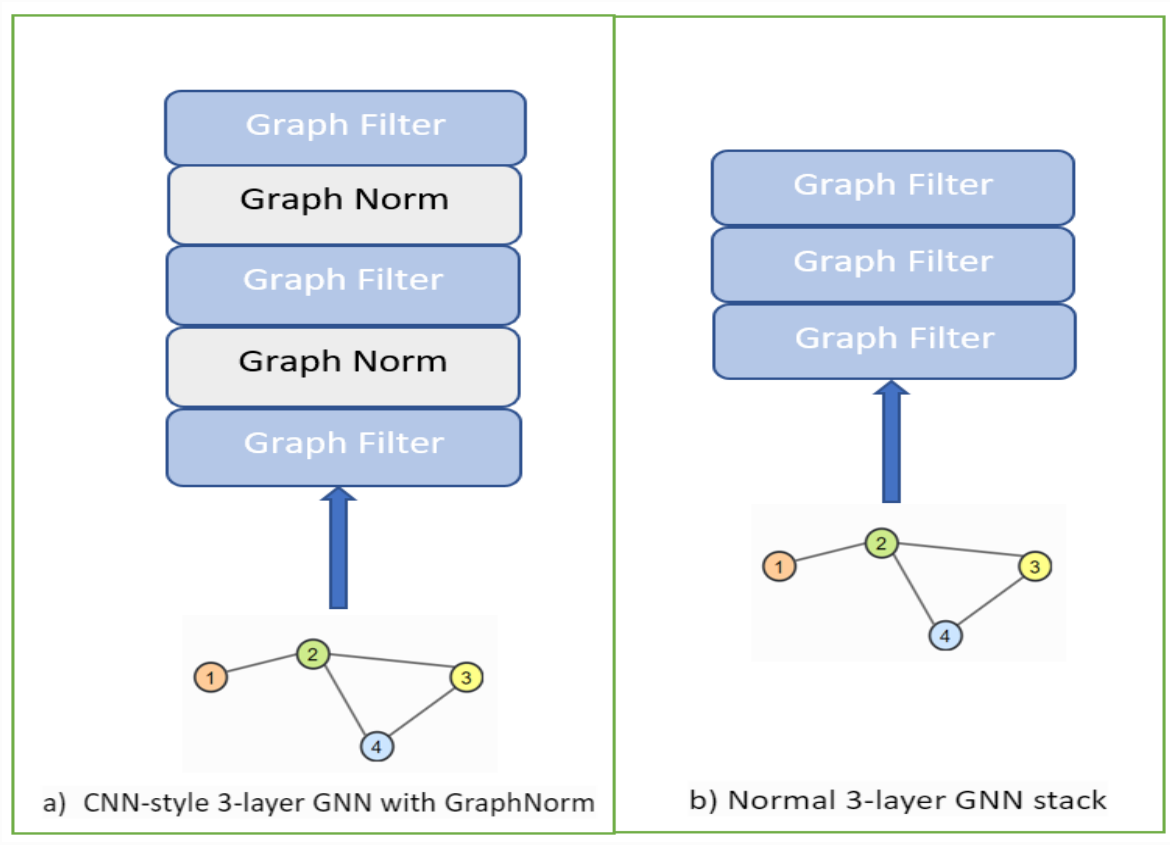
The dataset is already provided with necessary train, valid and test splits. They split the data according to time, in order to simulate a realistic application in collaboration recommendation. Specifically, they put the collaborations until 2017 as training edges, those in 2018 as validation edges, and those in 2019 as test edges. We are using both train and valid edges while evaluating the test set (transductive setting).

Model Architecture:

➢ The architecture of our model mainly contains two modules, which are trained end-to-end:

- First is a 3-layer Graph Neural Network stack. For different experiments, we considered either **SAGEConv** or **GCNConv** as graph filters to learn the node features.
  - o SAGEConv is the GraphSAGE operator [1]
  - o GCNConv is the normal graph convolutional operator [2].
- The second module is a link prediction module which uses node features of a pair to predict whether there is a link between them. This link predictor module is a normal 3-layer Dense Neural Network which is a binary classifier.

➢ For the first module, we also considered using **GraphNorm** normalisation, which applies graph normalisation over graphs [3]. We are stacking graphNorm layers similar to how we arrange batchnorm layers in CNN (as shown in the figure below, picture (a) is with GraphNorm layers and picture (b) is without GraphNorm).



- ➢ The following are the setups for our experiments that we considered, to see whether the CNN-style normalization improves the performance of GNN stack:
  1) SAGEConv
  2) SAGEConv + GraphNorm
  3) GCNConv
  4) GCNConv + GraphNorm
- ➢ We are also using negative sampling while calculating the loss function. For evaluating performance on test split, we are considering both train and val split links (transductive setting).

Evaluation Metrics:

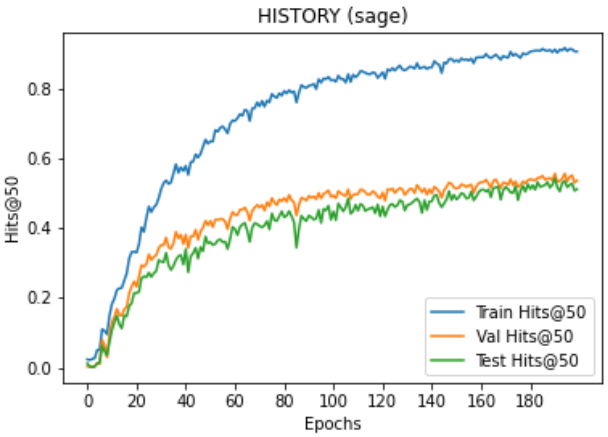
The leader board for the ogbl-collab dataset, considers **Hits@50** as the evaluation metric, which is defined as the fraction of positives that rank in the top 50 among their negatives (higher is better, best is 1). For example, let's assume that, in the top 50 collaborations, there are 30 positives and 20 negatives, then the Hits@50 for this situation is 30/50 = 0.6

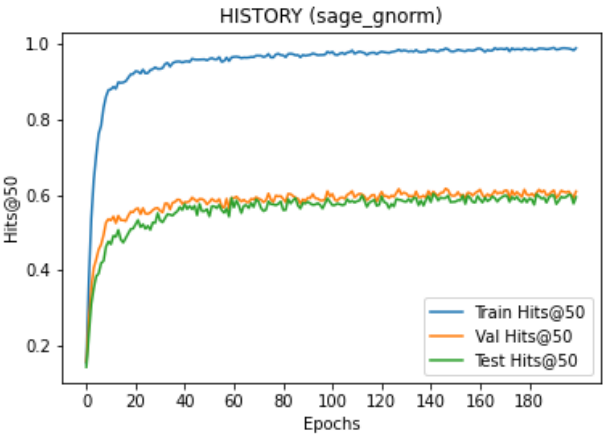
Results & Analysis:

The results for four different setups are as follows:

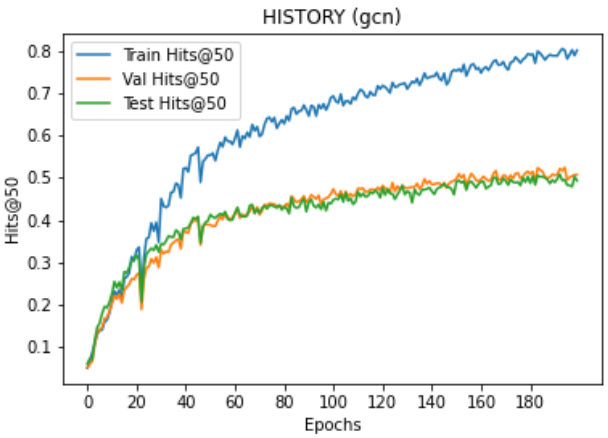
Setup	**Val Hits@50**	**Test Hits@50**
SAGEConv	0.5365	0.5113
SAGEConv + GraphNorm	0.6091	0.5935
GCNConv	0.5073	0.4935
GCNConv + GraphNorm	**0.6158**	**0.5991**

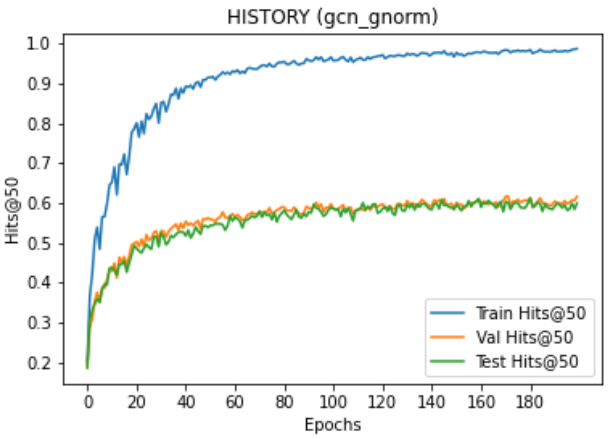
From the above table, we can observe that the best performing GNN stack is GCNConv





with GraphNorm layers. To understand the situation better, we used model training history to plot or visualize how Hits@50 value increased with the number of epochs. For both types of Graph Filters, the GNN stack with GraphNorm layers performed better. These GraphNorm stacks reached the peak just with a small number of epochs. But those without these GraphNorm layers took more epochs to reach their peak. The authors of [4] proposed GraphNorm with a learnable shift. Empirically, GNNs with GraphNorm converge faster compared to GNNs without normalization / using other forms of normalizations, this trend can be clearly observed in our experiments too.





Besides this, GraphNorm also improves the performance of GNN stacks on our link prediction task. Graph Normalization is analogous to Batch Normalization in Deep Neural Networks. These layers make sure that the outputs of a hidden layer don't scatter in big ranges. It standardizes the node embeddings obtained after each Graph Conv filter.

Conclusion:

From the above results and observations, we can conclude that for our setting, GCNConv with GraphNorm performs the best. On comparing our results with that in the ogbl-collab leaderboard, this approach stands somewhere between 10 to 12.

10	Common Neighbor	No	0.6137 ± 0.0000	0.6036 ± 0.0000
11	**SEAL-nofeat**	No	0.5471 ± 0.0049	0.6495 ± 0.0043
12	GraphSAGE (val as input)	No	0.5463 ± 0.0112	0.5688 ± 0.0077

References:

[1] Inductive Representation Learning on Large Graphs

[2] Semi-supervised Classification with Graph Convolutional Networks

[3] torch-geometric # normalization-layers

[4] GraphNorm: A Principled Approach to Accelerating Graph Neural Network Training