GNN Node Classification Task 01 Report

Sina Ziaee

The task is to predict the field of studies in the OBGN-Mag dataset.

Dataset summary:

No of papers: 736389
No of fields of study: 59965
No papers associated with fields: 7505078
No citations: 5416271
No affiliations: 1043998

Important fields used in the code(do not run this:

study_fields: list of all fields to all papers that they are associated with
study_fields_list: list of the number of papers that each field is associated with
papers_dict: a dictionary that contains the fields each paper has
field_of_study_with_max_papers: the field of study that is used in all papers
study_fields_dict: a dictionary that contains the number of papers for each paper
sorted_dict: a sorted dictionary of study_fields_dict dictionary
papers_study_fields: a list of all papers that contains the field of study which is associated with the highest number of papers
new_y: the torch tensor that is used to replace the data.y field

To perform a single-class single-label node classification, these actions were performed:

- Creating a vector that includes all fields of study. An edge type called " (paper, has_topic, field_of_study)" is available, where we can extract all the fields for each specific paper.
- Finding the number of papers associated with each field of study.
- creating a dictionary of papers and their corresponding fields of studies
- In the list where each element indicates the number of papers that a field is associated with, these are the number of papers for each field:
  [87464 129205 96621 86355 70668 67323 189963 **736389** 71231]
  As we can see, one field of study is associated with all of the papers, and this field is 14055.
- The overall analysis of the dataset suggests:

16758 fields of study are used only in one paper. Field of study "14055" is used in all papers. 3 papers only have 14055 as their only field of study.

We will select the fields of study that are used in the most number of papers. we will filter them based on fields that are used higher than 10000, 20000, ..., 50000 papers.

- Visualizing the number of fields and the papers is a plot.

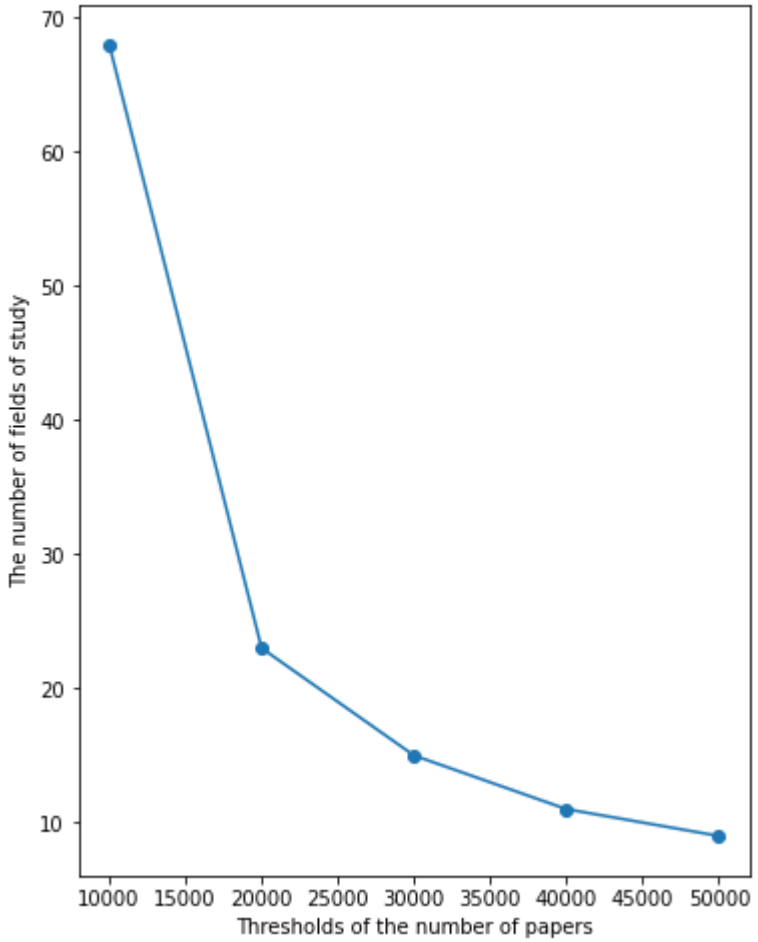


Fig 1) A visualization of papers and number of fields.
When the number of papers that has a specific field of study Increases the number of fields that have those number of papers decrease. Note that we didn't visualize the threshold of the number of papers with less than 10000 papers so that the differences would be obvious.

- Creating a list that, for each paper, returns the field of study associated with the highest number of papers. Note that we only use 14055 for a paper only and only if a particular paper contains only one field of study, which is 14055, so 3 papers will take this as their field of study with the highest number of associated papers.

This suggests that, in the end, 811 fields of studies are assigned to each paper.

Pipelines:

We have 4 different pipelines to implement GCN, GraphSage, GraphSaint, and RGCN.

From now on the pipelines used are the same as the examples on this github repo, but a brief introduction would be explained:

```
model.train() #

optimizer.zero_grad()             # zeroing optimizer for next epoch training
out = model(data.x, data.adj_t)[train_idx]      # model training

loss = F.nll_loss(out, data.y.squeeze(1)[train_idx])      # computing loss
loss.backward()          # backward operation (deravitive operations or gradient computation)
optimizer.step()         # performs a parameter update based on the current gradient

return loss.item()       # giveing the value of loss (out of the optimizer)
```

Loss function: Negative log likelihood loss

Optimizer: Adam
Learning rate: 0.01
Dropout_rate: 0.5
Number of input channels: (node features)=128
Number of output channels: 811 = (the number of fields of studies)
Number of hidden channels: 256, 128, or 64 depending on the model

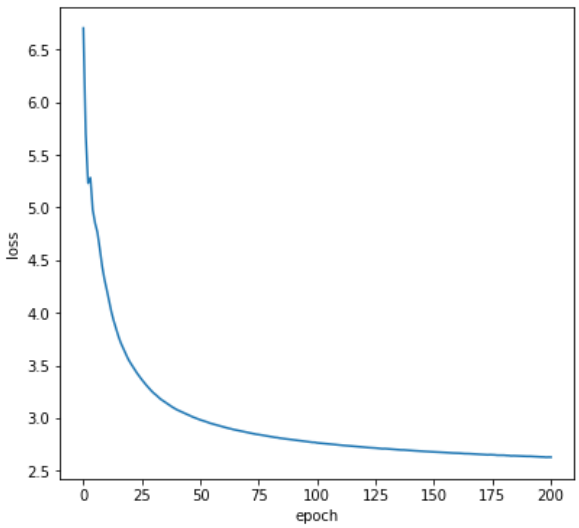
The data type that we created is Homogeneous data from the heterogeneous graph of papers, institutions, fields of study, and authors

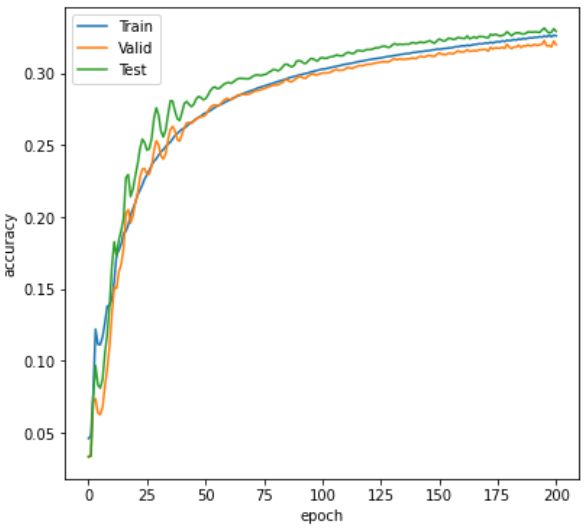
Results:

GraphSage results:

Run: 01, Epoch: 00, Loss: 6.7062, Train: 4.59%, Valid: 3.29% Test: 3.35% F1-Score: 3.35%
Stats(time=1.150097900390625, max_allocated_cuda=7895.96, max_reserved_cuda=8356.0, max_active_cuda=7895.96, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 20, Loss: 3.5278, Train: 20.99%, Valid: 21.02% Test: 22.85% F1-Score: 22.85%
Stats(time=1.1304151611328126, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 40, Loss: 3.0785, Train: 26.10%, Valid: 25.76% Test: 27.27% F1-Score: 27.27%
Stats(time=1.259049072265625, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 60, Loss: 2.9150, Train: 28.12%, Valid: 28.19% Test: 29.35% F1-Score: 29.35%
Stats(time=1.20804541015625, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 80, Loss: 2.8234, Train: 29.37%, Valid: 29.16% Test: 30.23% F1-Score: 30.23%
Stats(time=1.2321156005859375, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 100, Loss: 2.7637, Train: 30.28%, Valid: 30.00% Test: 31.08% F1-Score: 31.08%
Stats(time=1.2177252197265624, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 120, Loss: 2.7230, Train: 30.94%, Valid: 30.62% Test: 31.63% F1-Score: 31.63%
Stats(time=1.2436971435546875, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 140, Loss: 2.6910, Train: 31.44%, Valid: 31.00% Test: 32.07% F1-Score: 32.07%
Stats(time=1.2217769775390626, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 160, Loss: 2.6661, Train: 31.91%, Valid: 31.46% Test: 32.43% F1-Score: 32.43%
Stats(time=1.2303690185546876, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 180, Loss: 2.6457, Train: 32.25%, Valid: 31.81% Test: 32.75% F1-Score: 32.75%
Stats(time=1.22181201171875, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)
Run: 01, Epoch: 200, Loss: 2.6276, Train: 32.60%, Valid: 31.98% Test: 32.91% F1-Score: 32.91%
Stats(time=1.2227042236328125, max_allocated_cuda=7901.48, max_reserved_cuda=8358.0, max_active_cuda=7901.48, nvidia_smi_free_cuda=1689.13, nvidia_smi_used_cuda=14152.65)

Training time: 392.8833293914795s (for 201 epochs)
Memory used: 7901.48 MB





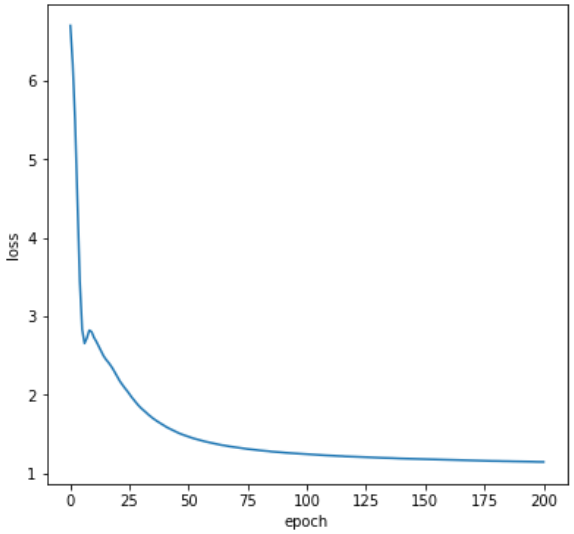
As it is shown, model is underfitting. By adding to layers and number of channels in each layer, we can achieve a more complicated model with higher performance. Unfortunately, it is not possible on colab with free account as it exceeds in memory.

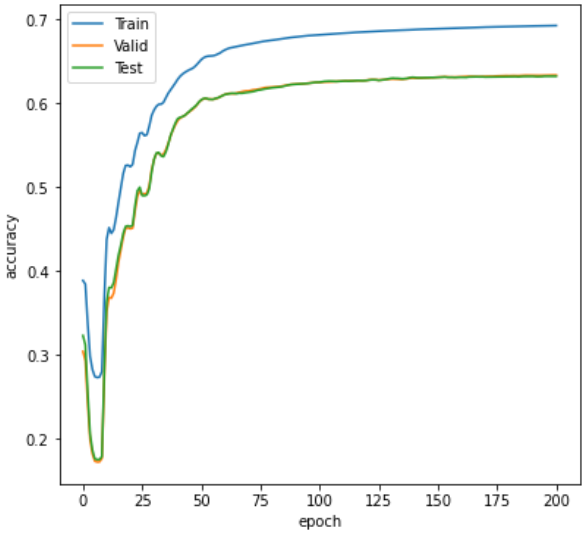
GCN results:

Run: 01, Epoch: 00, Loss: 6.7018, Train: 38.84%, Valid: 30.38% Test: 32.27% F1-Score: 32.27%
Stats(time=1.3770419921875, max_allocated_cuda=8700.64, max_reserved_cuda=9074.0, max_active_cuda=8700.64, nvidia_smi_free_cuda=948.89, nvidia_smi_used_cuda=14892.89)
Run: 01, Epoch: 20, Loss: 2.2167, Train: 52.43%, Valid: 45.02% Test: 45.29% F1-Score: 45.29%
Stats(time=1.3010291748046876, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 40, Loss: 1.5989, Train: 62.86%, Valid: 57.92% Test: 58.16% F1-Score: 58.16%
Stats(time=1.26175390625, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 60, Loss: 1.3811, Train: 66.38%, Valid: 61.06% Test: 61.06% F1-Score: 61.06%
Stats(time=1.3159833984375, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 80, Loss: 1.2895, Train: 67.54%, Valid: 61.95% Test: 61.86% F1-Score: 61.86%
Stats(time=1.3188399658203125, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 100, Loss: 1.2404, Train: 68.18%, Valid: 62.50% Test: 62.52% F1-Score: 62.52%
Stats(time=1.26237890625, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 120, Loss: 1.2091, Train: 68.54%, Valid: 62.75% Test: 62.75% F1-Score: 62.75%
Stats(time=1.3093028564453124, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 140, Loss: 1.1858, Train: 68.80%, Valid: 63.02% Test: 63.10% F1-Score: 63.10%
Stats(time=1.265591552734375, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 160, Loss: 1.1689, Train: 69.01%, Valid: 63.17% Test: 63.11% F1-Score: 63.11%
Stats(time=1.2588365478515624, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 180, Loss: 1.1537, Train: 69.18%, Valid: 63.29% Test: 63.21% F1-Score: 63.21%
Stats(time=1.313676025390625, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)
Run: 01, Epoch: 200, Loss: 1.1425, Train: 69.28%, Valid: 63.36% Test: 63.23% F1-Score: 63.23%
Stats(time=1.3072398681640625, max_allocated_cuda=8702.03, max_reserved_cuda=8992.0, max_active_cuda=8702.03, nvidia_smi_free_cuda=760.16, nvidia_smi_used_cuda=15081.62)

Training time: 422.76872301101685s

Memory Used: 8702.03 MB





As it is shown, the model is performing better than GraphSage, but it in my opinion we can still get higher performance with more complicated model. However, we must consider the probability of overfitting as some signatures about this fact is available here(6% difference with train and test accuracies)

RGCN and GraphSaint results:

Unfortunately, google colab free account does not support higher memory and the model exceeds the amount of allowed memory to be used.

For RGCN model:

RuntimeError: CUDA out of memory. Tried to allocate 3.43 GiB (GPU 0; 14.76 GiB total capacity; 11.82 GiB already allocated; 2.02 GiB free; 11.92 GiB reserved in total by PyTorch).

For GraphSaint model:

RuntimeError: CUDA out of memory. Tried to allocate 2.23 GiB (GPU 0; 14.76 GiB total capacity; 10.18 GiB already allocated; 1.58 GiB free; 12.37 GiB reserved in total by PyTorch).

Please take into account that using less number of classes in data.y could result in higher accuracies. Such as if we used top 20 fields of studies instead of finding the field of study for each paper that is associated with most number of papers, the result could be significantly better. Unfortunately, the time limitations and other tasks that I have finish, do not allow me to work on these aspects for the time beeing. Maybe in the future.

Thank you for taking your time and considering these results.