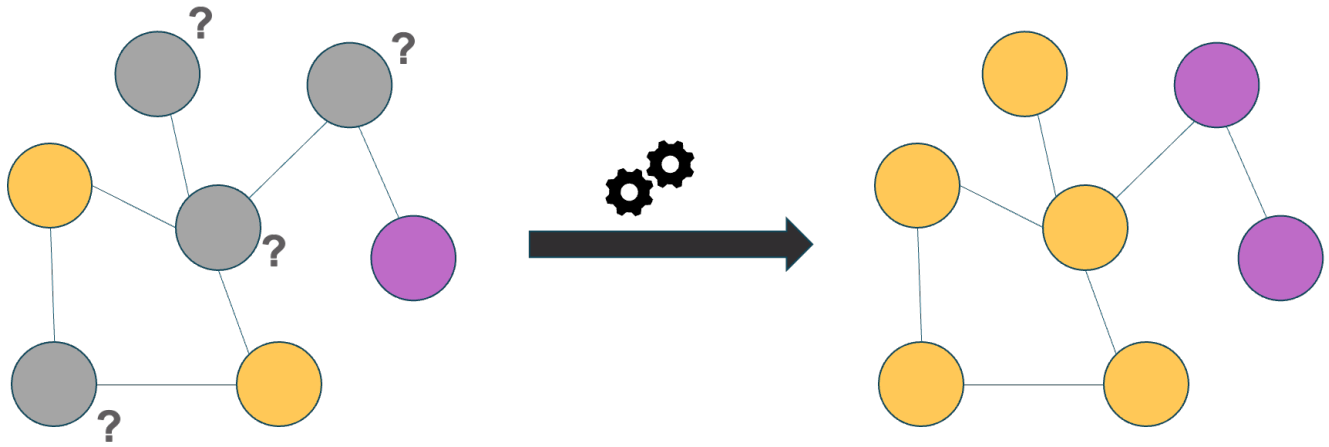
Data Science HW3

Department of Computer Science
National Tsing Hua University (NTHU)
Hsinchu, Taiwan

Due Date: 2024/05/14 (Tue) 23:59

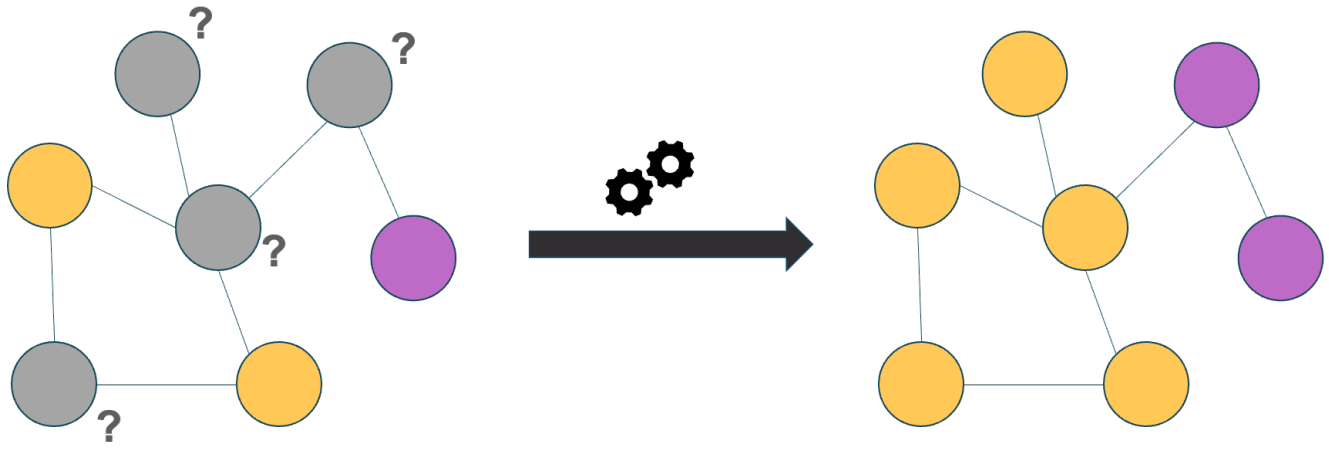
TA : 呂佳勳 資電館743
Email: lobsterlab.cs.nthu@gmail.com

HW3 Node classification



- Build a node classification model
- Given an unknown graph dataset
  - Train your model using the training nodes
  - Predict the labels of the testing nodes.

HW3 Semi-Supervised Learning



- Use unlabeled data and labeled data to train model.
- #unlabeled data >> #labeled data

HW3 is hosted on Kaggle

- HW 3 Kaggle link
  - https://www.kaggle.com/t/623509cc8ae648cc85480bfe365d895a
  - Deadline: 2023/05/14 (Tue) 23:59

- **Fill your Kaggle name in the google form**
- We will **use the result on Kaggle to score** this homework

Dataset description (1/2)

- A modified graph data
  - Each node has a predefined feature.
- Dataset file name description

```
dataset
|    ├── private_features.pkl # node feature
|    ├── private_graph.pkl # graph edges
|    ├── private_num_classes.pkl # number of classes
|    ├── private_test_labels.pkl # X
|    ├── private_test_mask.pkl # nodes indices of testing set
|    ├── private_train_labels.pkl # nodes labels of training set
|    ├── private_train_mask.pkl # nodes indices of training set
|    ├── private_val_labels.pkl # nodes labels of validation set
|    └── private_val_mask.pkl # nodes indices of validation set
```

Dataset description (2/2)

- **Why training data size is far less than validation and testing data?**
  - The purpose of splitting the data in this way is to test the learning ability of the graph neural network when there is a lack of labeled data.
  - In such a situation, it can be difficult to use other machine learning models that do not take structural information into account.
  - Please do not modify the training and validation data.

- **Why #training data + #validation data + #testing data != #total nodes(60+600+1200!=20000) ?**
  - During the training of a graph neural network, the model constructs learned representations using features from neighboring nodes. As a result, even nodes that are not part of the training, validation, or test sets can still have an impact on the learned representation.
  - The setting is followed other frequently used public datasets.

Output format

- For each testing instance, there is a unique id
- Output your prediction to csv file with the following format and submit to Kaggle
- **Remember to write the first line 'ID,Predict'**
- Output csv file example:

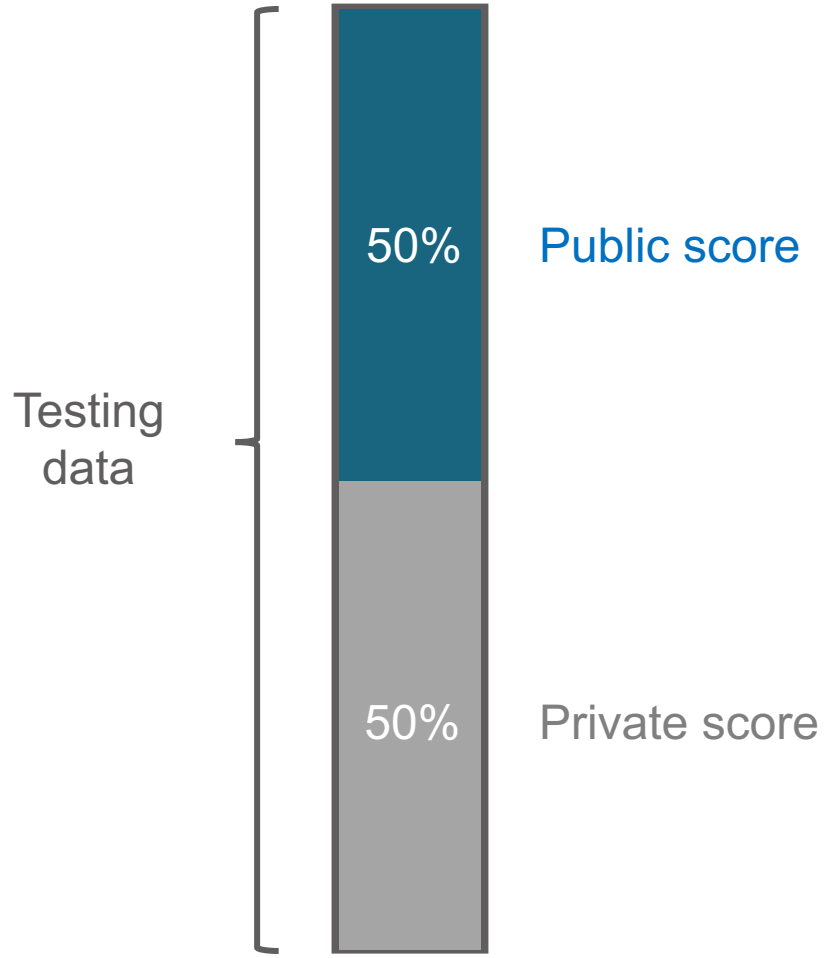
1	ID	Predict
2	0	1
3	1	0
4	2	0
5	3	2
6	4	0
7	5	2
8	6	2
9	7	2
10	8	0

Evaluation

- Using **Accuracy**
  - $\frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$
- There are two leaderboards on Kaggle
  - **Public**
    - Can be seen during competition
  - **Private**
    - Can be seen after competition

Public and Private leaderboard

- **Public** (Can be seen during competition)
  - 50% testing data
  - For reference
- **Private** (Can be seen after competition)
  - the other 50%
  - **Use this result for final scoring**



Scoring

	Public	Private*	Your HW3 Score
Baseline 0	0.729	**0.718**	0
Baseline 60	0.764	**0.765**	60
Baseline 70	0.788	**0.799**	70
Baseline 80	0.818	**0.819**	80
Baseline 88	0.838	**0.840**	Top 20% : 100 20%~80%: 92 Others: 88

- You will get **0**, if your **private score** is between *baseline 0* and *baseline 60*
- You will get **60**, if your **private score** is between *baseline 60* and *baseline 70*
- You will get **70**, if your **private score** is between *baseline 70* and *baseline 80*
- And so on

** The private score of each baseline may be adjusted based on the results of classmates.*

Scoring

- Baseline scores
  - There are benchmarks on the leaderboard for reference

#	Team	Members	Score	Entries	Last
	baseline_88.csv		0.83828		
	baseline_80.csv		0.81848		
	baseline_70.csv		0.78877		
	baseline_60.csv		0.76402		
	baseline_0.csv		0.72937		

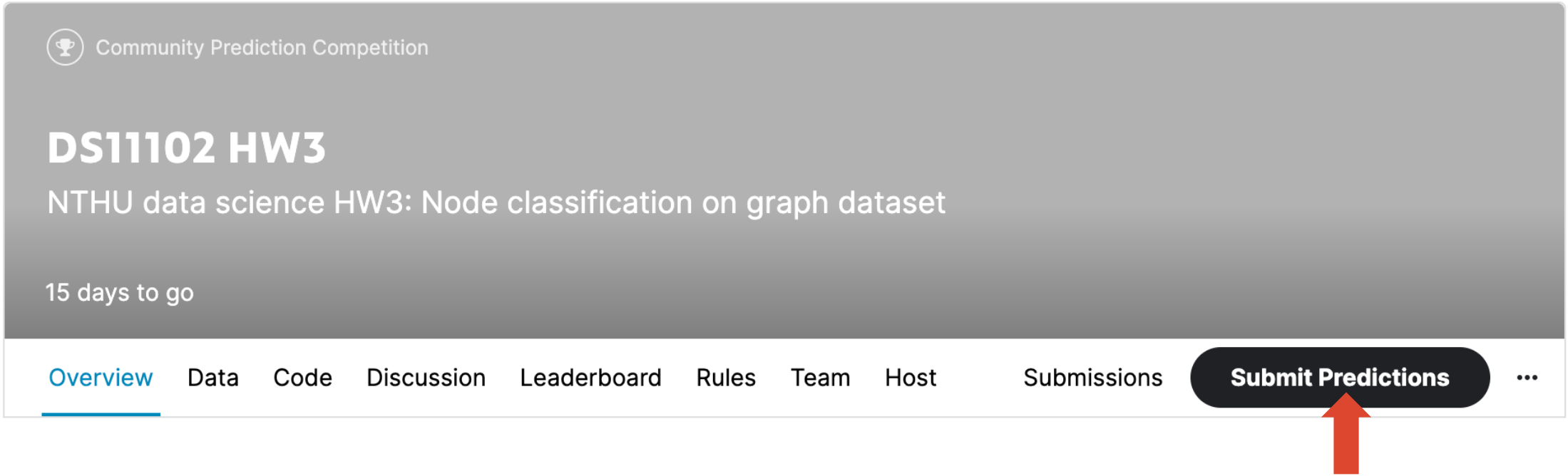
Other rules

- You can submit 20 times per day
- You can choose 4 predictions for final scoring
  - Kaggle will use the best one to be your result

- We will publish private leaderboard one time on eeclass 3 days before the deadline of this homework.
  - This won't affect your final score, just for you to check your status.

How to submit

- Click ***'Submit Predictions'*** button on the navigation bar



How to submit

Submit to Competition

File Upload | Notebook

DS11102 HW3
You have 19 submissions remaining today. This resets in an hour.

Drag and drop file to upload
(e.g., .csv, .zip, .gz, .7z)

Upload your answer csv file here

or

Browse Files

Your submission should be a CSV file with 1000 rows and a header. You can upload a zip/gz/7z archive.

DESCRIPTION

Enter a description

You can write some description about the answer csv file

0 / 500

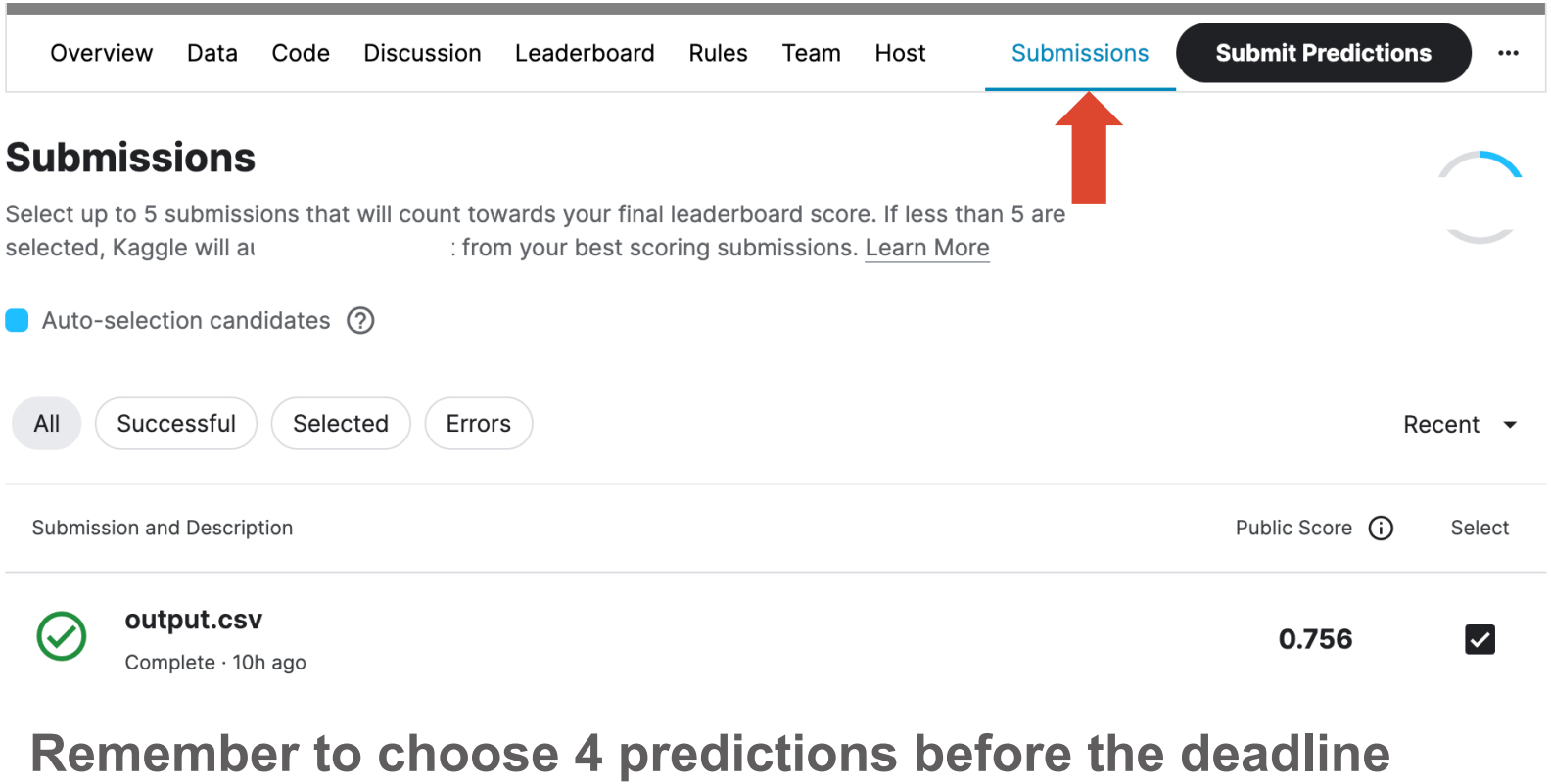
```
>_ kaggle competitions submit -c ds11102-hw3 -f submission.csv -m …
```

Cancel **Submit**

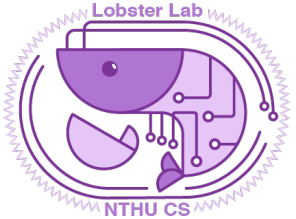
Click to submit

Choose predictions for final scoring

- You can see all your submissions in ***'Submissions'***



Remember to choose 4 predictions before the deadline



Sample code description

Sample code

- Load data based on provided dataset name.
- No need to modify this file.

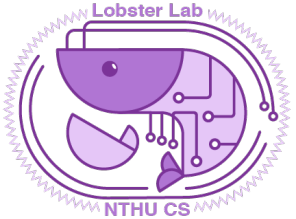
- Define the details of model.

- Train and evaluate the model.
- Export the submission csv file.

data_loader.py

```
def load_data():
    """
    * Load data from pickle file in folder `dataset`.
    * No need to modify.

    * test_labels is an array of length 1200 with each element being -1.
    * train_mask, val_mask, and test_mask are used to indicate the index of each set of nodes.
    """
```



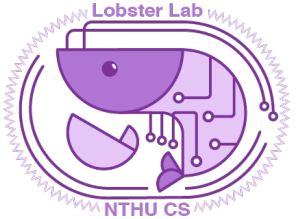
train.py

```
78      # Load data
79      features, graph, num_classes, \
80      train_labels, val_labels, test_labels, \
81      train_mask, val_mask, test_mask = load_data()
82
83      # Initialize the model (Baseline Model: GCN)
84      """TODO: build your own model in model.py and replace GCN() with your model"""
85      in_size = features.shape[1]
86      out_size = num_classes
87      model = GCN(in_size, 16, out_size).to(device)
```

model.py

```python
class GCN(nn.Module):
    """
    Baseline Model:
    - A simple two-layer GCN model, similar to https://github.com/tkipf/pygcn
    - Implement with DGL package
    """
    def __init__(self, in_size, hid_size, out_size):
        super().__init__()
        self.layers = nn.ModuleList()
        # two-layer GCN
        self.layers.append(
            GraphConv(in_size, hid_size, activation=F.relu)
        )
        self.layers.append(GraphConv(hid_size, out_size))
        self.dropout = nn.Dropout(0.5)

    def forward(self, g, features):
        h = features
        for i, layer in enumerate(self.layers):
            if i != 0:
                h = self.dropout(h)
            h = layer(g, h)
        return h
```



model.py

```
30  # class YourGNNModel(nn.Module):
31  #     """
32  #     TODO: Use GCN model as reference, implement your own model here to achieve higher accuracy on testing data
33  #     """
34  #     def __init__(self, in_size, hid_size, out_size):
35  #         super().__init__()
36  #
37  #     def forward(self, g, features):
38  #         pass
39
```

How to run

- Run:
- `python3 train.py \`
  `--epochs {num of epochs} \`
  `--es_iters {num of iters to trigger early stopping}\`
  `--use-gpu`

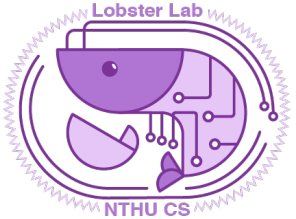
- Example:

```
python3 train.py --es_iters 30 --epochs 300 --use_gpu
```

Once you successfully run the sample code…

```
Training...
Early stopping monitoring on
Epoch 00000 | Loss 1.1011 | Accuracy 0.1980
Epoch 00001 | Loss 1.0950 | Accuracy 0.2140
Epoch 00002 | Loss 1.0918 | Accuracy 0.5120
Epoch 00003 | Loss 1.0873 | Accuracy 0.5920
Epoch 00004 | Loss 1.0790 | Accuracy 0.6120
Epoch 00005 | Loss 1.0724 | Accuracy 0.6340
Epoch 00006 | Loss 1.0644 | Accuracy 0.6500
Epoch 00007 | Loss 1.0541 | Accuracy 0.6860
Epoch 00008 | Loss 1.0476 | Accuracy 0.7100
Epoch 00009 | Loss 1.0333 | Accuracy 0.7240
Epoch 00010 | Loss 1.0253 | Accuracy 0.7180
......
Epoch 00155 | Loss 0.2243 | Accuracy 0.7900
Epoch 00156 | Loss 0.2472 | Accuracy 0.7880
Epoch 00157 | Loss 0.1898 | Accuracy 0.7900
Epoch 00158 | Loss 0.1982 | Accuracy 0.7880
Epoch 00159 | Loss 0.2302 | Accuracy 0.7840
Epoch 00160 | Loss 0.1814 | Accuracy 0.7860
Early stopping at epoch=161
Testing...
Export predictions as csv file.
```

* Please remember to upload your output csv file to Kaggle for scoring.



HW3 Conclusion

- Submit your code to eeclass before deadline
  1. All your python scripts file which is able to read dataset, train your model, export a prediction csv file
  2. a README.md file
     - describe how to run your code
  - Your code must be able to show that the predictions are derived from a machine learning or deep learning model.
- Evaluate your model performance on Kaggle.
  - Submit your model prediction file
  - We will **use the result on Kaggle to score** this homework.

For Your Reference

- DGL: https://docs.dgl.ai/index.htm

- GAT: https://arxiv.org/pdf/1710.10903v3.pdf
- SSP: https://arxiv.org/pdf/2008.09624v1.pdf
- GRACE: https://arxiv.org/abs/2006.04131v2
- State-of-the-art on Node Classification : https://paperswithcode.com/task/node-classification