GraphNNs Practice
Installation Instructions

FOR ALL:
    source /opt/anaconda/bin/activate root
    conda deactivate

CPU/GPU (for cuda 10; adapt the command to your own config):
    conda create -n graphnn_gpu_env python=3.8 anaconda -y && \
    conda activate graphnn_gpu_env && \
    conda install pytorch=1.7.1 cudatoolkit=10.2 -c pytorch -y && \
    conda install -c dglteam dgl-cuda10.2 -y

CPU ONLY (should work in all cases):
    conda create -n graphnn_cpu_env python=3.8 anaconda -y && \
    conda activate graphnn_cpu_env && \
    conda install -c pytorch pytorch-cpu -y && \
    conda install -c dglteam dgl -y
Description of the dataset

20 graphs for training, 2 graphs for test
2372 nodes on average per graph
Each node has 50 features and 121 labels

The task

- Take the code that has been provided and improve it (most improvements will come from the architecture) (using a Graph Attention Network is highly recommended) → This counts for 8/20 points

- Produce a diagram of the architecture that you’re using (shape information must be included) and explaining the difference between:
  * similarity attention (defined in https://arxiv.org/abs/1706.03762 (Vaswani et al.))
  * utility attention (defined in https://arxiv.org/abs/1710.10903 (Veličković et al.))
Equations and brief mechanism explanation are expected! → This counts for 12/20 points
The task

- In summary, you have 4 files to produce:
  → the modified code (train_ppi.py)
  → the weights of the model (model_state.pth)
  → the diagram of the model (an image) with a summary about difference between utility and similarity attention

- Do not change the signature of the train() and test() functions!

Make sure your submission correctly runs within your conda environment with:

```bash
python3 train_ppi.py --mode test
```
Diagram (perfectible) examples

Multi-head attention
(shape information is missing)

ResNeXt
(a good legend is missing)
Important Hints

Use a Graph Attention Network (Veličković et al.):
https://arxiv.org/abs/1710.10903

Use the examples from the DGL dependency:
https://github.com/dmlc/dgl
https://www.dgl.ai/

Use inspiration from pytorch geometric (but don’t use it directly):
https://github.com/rusty1s/pytorch_geometric