In [1]:
```
import dgl
import dgl.function as fn
import torch as th
import torch.nn as nn
import torch.nn.functional as F
from dgl import DGLGraph

gcn_msg = fn.copy_src(src='h', out='m')
gcn_reduce = fn.sum(msg='m', out='h')
```

In [2]:
```
class NodeApplyModule(nn.Module):
    def __init__(self, in_feats, out_feats, activation):
        super(NodeApplyModule, self).__init__()
        self.linear = nn.Linear(in_feats, out_feats)
        self.activation = activation

    def forward(self, node):
        h = self.linear(node.data['h'])
        if self.activation is not None:
            h = self.activation(h)
        return {'h' : h}
```

In [3]:
```
class GCN(nn.Module):
    def __init__(self, in_feats, out_feats, activation):
        super(GCN, self).__init__()
        self.apply_mod = NodeApplyModule(in_feats, out_feats, activation)

    def forward(self, g, feature):
        g.ndata['h'] = feature
        g.update_all(gcn_msg, gcn_reduce)
        g.apply_nodes(func=self.apply_mod)
        return g.ndata.pop('h')
```

In [4]:
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.gcn1 = GCN(1433, 16, F.relu)
        self.gcn2 = GCN(16, 7, None)

    def forward(self, g, features):
        x = self.gcn1(g, features)
        x = self.gcn2(g, x)
        return x
net = Net()
print(net)
```

```
Net(
  (gcn1): GCN(
    (apply_mod): NodeApplyModule(
      (linear): Linear(in_features=1433, out_features=16, bias=True)
    )
  )
  (gcn2): GCN(
    (apply_mod): NodeApplyModule(
      (linear): Linear(in_features=16, out_features=7, bias=True)
    )
  )
)
```

In [5]:
```
from dgl.data import citation_graph as citegrh
import networkx as nx
def load_cora_data():
    data = citegrh.load_cora()
    features = th.FloatTensor(data.features)
    labels = th.LongTensor(data.labels)
    train_mask = th.BoolTensor(data.train_mask)
    test_mask = th.BoolTensor(data.test_mask)
    g = data.graph
    # add self loop
    g.remove_edges_from(nx.selfloop_edges(g))
    g = DGLGraph(g)
    g.add_edges(g.nodes(), g.nodes())
    return g, features, labels, train_mask, test_mask
```

In [6]:
```
def evaluate(model, g, features, labels, mask):
    model.eval()
    with th.no_grad():
        logits = model(g, features)
        logits = logits[mask]
        labels = labels[mask]
        _, indices = th.max(logits, dim=1)
        correct = th.sum(indices == labels)
        return correct.item() * 1.0 / len(labels)
```

In [7]:
```
import time
import numpy as np
g, features, labels, train_mask, test_mask = load_cora_data()
optimizer = th.optim.Adam(net.parameters(), lr=1e-3)
dur = []
for epoch in range(50):
    if epoch >=3:
        t0 = time.time()

    net.train()
    logits = net(g, features)
    logp = F.log_softmax(logits, 1)
    loss = F.nll_loss(logp[train_mask], labels[train_mask])

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if epoch >=3:
        dur.append(time.time() - t0)

    acc = evaluate(net, g, features, labels, test_mask)
    print("Epoch {:05d} | Loss {:.4f} | Test Acc {:.4f} | Time(s) {:.4f}".form
at(
            epoch, loss.item(), acc, np.mean(dur)))
```

```
C:\Users\qiqishaoshuai\Miniconda3\lib\site-packages\numpy\core\fromnumeric.p
y:3335: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
C:\Users\qiqishaoshuai\Miniconda3\lib\site-packages\numpy\core\_methods.py:16
1: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)

Epoch 00000 | Loss 1.9850 | Test Acc 0.1510 | Time(s) nan
Epoch 00001 | Loss 1.9659 | Test Acc 0.1550 | Time(s) nan
Epoch 00002 | Loss 1.9495 | Test Acc 0.1640 | Time(s) nan
Epoch 00003 | Loss 1.9341 | Test Acc 0.2250 | Time(s) 0.0628
Epoch 00004 | Loss 1.9185 | Test Acc 0.2520 | Time(s) 0.0643
Epoch 00005 | Loss 1.9020 | Test Acc 0.2700 | Time(s) 0.0645
Epoch 00006 | Loss 1.8846 | Test Acc 0.2910 | Time(s) 0.0666
Epoch 00007 | Loss 1.8664 | Test Acc 0.3070 | Time(s) 0.0686
Epoch 00008 | Loss 1.8475 | Test Acc 0.3180 | Time(s) 0.0705
Epoch 00009 | Loss 1.8284 | Test Acc 0.3260 | Time(s) 0.0704
Epoch 00010 | Loss 1.8090 | Test Acc 0.3350 | Time(s) 0.0702
Epoch 00011 | Loss 1.7891 | Test Acc 0.3500 | Time(s) 0.0708
Epoch 00012 | Loss 1.7691 | Test Acc 0.3610 | Time(s) 0.0714
Epoch 00013 | Loss 1.7488 | Test Acc 0.3590 | Time(s) 0.0725
Epoch 00014 | Loss 1.7287 | Test Acc 0.3650 | Time(s) 0.0722
Epoch 00015 | Loss 1.7088 | Test Acc 0.3760 | Time(s) 0.0730
Epoch 00016 | Loss 1.6895 | Test Acc 0.3790 | Time(s) 0.0728
Epoch 00017 | Loss 1.6702 | Test Acc 0.3900 | Time(s) 0.0729
Epoch 00018 | Loss 1.6506 | Test Acc 0.3950 | Time(s) 0.0725
Epoch 00019 | Loss 1.6309 | Test Acc 0.4000 | Time(s) 0.0740
Epoch 00020 | Loss 1.6115 | Test Acc 0.4080 | Time(s) 0.0737
Epoch 00021 | Loss 1.5924 | Test Acc 0.4130 | Time(s) 0.0740
Epoch 00022 | Loss 1.5741 | Test Acc 0.4220 | Time(s) 0.0736
Epoch 00023 | Loss 1.5559 | Test Acc 0.4240 | Time(s) 0.0734
Epoch 00024 | Loss 1.5378 | Test Acc 0.4310 | Time(s) 0.0731
Epoch 00025 | Loss 1.5199 | Test Acc 0.4370 | Time(s) 0.0728
Epoch 00026 | Loss 1.5021 | Test Acc 0.4470 | Time(s) 0.0725
Epoch 00027 | Loss 1.4844 | Test Acc 0.4560 | Time(s) 0.0725
Epoch 00028 | Loss 1.4670 | Test Acc 0.4750 | Time(s) 0.0724
Epoch 00029 | Loss 1.4496 | Test Acc 0.4930 | Time(s) 0.0723
Epoch 00030 | Loss 1.4324 | Test Acc 0.5040 | Time(s) 0.0722
Epoch 00031 | Loss 1.4153 | Test Acc 0.5150 | Time(s) 0.0720
Epoch 00032 | Loss 1.3982 | Test Acc 0.5310 | Time(s) 0.0718
Epoch 00033 | Loss 1.3808 | Test Acc 0.5470 | Time(s) 0.0718
Epoch 00034 | Loss 1.3635 | Test Acc 0.5590 | Time(s) 0.0718
Epoch 00035 | Loss 1.3465 | Test Acc 0.5610 | Time(s) 0.0716
Epoch 00036 | Loss 1.3297 | Test Acc 0.5710 | Time(s) 0.0716
Epoch 00037 | Loss 1.3132 | Test Acc 0.5790 | Time(s) 0.0720
Epoch 00038 | Loss 1.2968 | Test Acc 0.5890 | Time(s) 0.0721
Epoch 00039 | Loss 1.2806 | Test Acc 0.5960 | Time(s) 0.0721
Epoch 00040 | Loss 1.2644 | Test Acc 0.6080 | Time(s) 0.0725
Epoch 00041 | Loss 1.2483 | Test Acc 0.6190 | Time(s) 0.0725
Epoch 00042 | Loss 1.2324 | Test Acc 0.6320 | Time(s) 0.0726
Epoch 00043 | Loss 1.2167 | Test Acc 0.6450 | Time(s) 0.0726
Epoch 00044 | Loss 1.2011 | Test Acc 0.6610 | Time(s) 0.0727
Epoch 00045 | Loss 1.1858 | Test Acc 0.6730 | Time(s) 0.0726
Epoch 00046 | Loss 1.1706 | Test Acc 0.6770 | Time(s) 0.0725
Epoch 00047 | Loss 1.1556 | Test Acc 0.6830 | Time(s) 0.0725
Epoch 00048 | Loss 1.1406 | Test Acc 0.6870 | Time(s) 0.0724
Epoch 00049 | Loss 1.1258 | Test Acc 0.6940 | Time(s) 0.0723
```

In [ ]: