```
!pip install torch_geometric

Collecting torch_geometric
  Downloading torch_geometric-2.6.1-py3-none-any.whl.metadata (63 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/63.1 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.1/63.1 kB 2.0 MB/s eta
0:00:00
ent already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from torch_geometric) (3.11.13)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.11/dist-packages (from torch_geometric)
(2024.10.0)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.11/dist-packages (from torch_geometric) (3.1.6)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (from torch_geometric) (2.0.2)
Requirement already satisfied: psutil>=5.8.0 in
/usr/local/lib/python3.11/dist-packages (from torch_geometric) (5.9.5)
Requirement already satisfied: pyparsing in
/usr/local/lib/python3.11/dist-packages (from torch_geometric) (3.2.1)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from torch_geometric)
(2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (from torch_geometric) (4.67.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp-
>torch_geometric) (1.18.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch_geometric)
(3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
```

```
/usr/local/lib/python3.11/dist-packages (from requests-
>torch_geometric) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests-
>torch_geometric) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests-
>torch_geometric) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests-
>torch_geometric) (2025.1.31)
Downloading torch_geometric-2.6.1-py3-none-any.whl (1.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 22.0 MB/s eta
0:00:00
etric
Successfully installed torch_geometric-2.6.1
```

```python
import numpy as np
import torch
import torch_geometric
import h5py
import matplotlib.pyplot as plt

from torch_geometric.data import Data, Batch
from torch_geometric.loader import DataLoader
from sklearn.neighbors import kneighbors_graph
from sklearn.model_selection import train_test_split

import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

from torch_geometric.nn import global_mean_pool
from torch.nn import Linear

Data_Path = '/content/drive/MyDrive/quark-gluon_data-set_n139306.hdf5'
Data_Size = 10000
k = 10

# The coords of the data in the 125 x 125 grid is appended to the data
# This is done to be able to use the data later for GAE model
data = h5py.File(Data_Path, "r")
images = data['X_jets'][0:Data_Size]
labels = data['y'][0:Data_Size]
coords = np.indices((125, 125))
coords = np.moveaxis(coords, 1, -1).T
coords = np.expand_dims(coords, axis=0)
# coords are normalized to be in the range [0, 1]
coords = coords.astype(np.float32) / 125.
coords = np.repeat(coords, 10000, axis=0)
```

```
# Some initial fetures were provided in the dataset m0 and pt which
are used as global features
# The global features are log transformed
global_feats = np.vstack((data['m0'][:Data_Size], data['pt']
[:Data_Size])).T
global_feats =torch.log1p(torch.from_numpy(global_feats))

images_with_coords = np.concatenate((images, coords), axis=-1)

del coords
del images
del data

# Reshape the data to be compatible with torch_geometric
data = images_with_coords.reshape((-1,
images_with_coords.shape[1]*images_with_coords.shape[2], 5))
non_black_pixels_mask = np.any(data[..., :3] != [0., 0., 0.], axis=-1)
del images_with_coords
node_list = []
for i, x in enumerate(data):
    node_list.append(x[non_black_pixels_mask[i]])


dataset = []
for i, nodes in enumerate(node_list):
    if i == 0:
      print(nodes.shape)
    edges = kneighbors_graph(nodes[...,3:], k, mode='connectivity',
include_self=True)
    c = edges.tocoo()
    edge_list = torch.from_numpy(np.vstack((c.row,
c.col))).type(torch.long)
    edge_weight = torch.from_numpy(c.data.reshape(-1, 1))
    y = torch.tensor([int(labels[i])], dtype=torch.long)
    data = Data(x=torch.from_numpy(nodes), edge_index=edge_list,
edge_attr=edge_weight, y=y, global_feat=global_feats[i,None])
    dataset.append(data)
```

```
(884, 5)
```

```
# The classifictaion model was traned on a subset of the full dataset
train_loader = DataLoader(dataset[:8000], batch_size=32, shuffle=True)
test_loader = DataLoader(dataset[8000:9000], batch_size=32,
shuffle=False)
val_loader = DataLoader(dataset[9000:], batch_size=32, shuffle=False)

from torch_geometric.nn import GCNConv

class GCNClassifier(torch.nn.Module):
  def __init__(self, input_embed_dim : int, latent_dim = None,
num_classes : int = 2):
```

```
    super(GCNClassifier, self).__init__()
    self.node_dim = input_embed_dim
    if latent_dim is None:
      self.latent_dim = self.node_dim
    else:
      self.latent_dim = latent_dim
    self.num_classes = num_classes
    self.Conv1 = GCNConv(self.node_dim, self.latent_dim)
    self.Conv2 = GCNConv(self.latent_dim, 2*self.latent_dim)
    self.Conv3 = GCNConv(2*self.latent_dim, 4*self.latent_dim)
    self.lin1 = torch.nn.Linear(4*self.latent_dim + 2,
2*self.latent_dim)
    self.lin2 = torch.nn.Linear(2*self.latent_dim, self.latent_dim)
    self.lin3 = torch.nn.Linear(self.latent_dim, 4*self.num_classes)
    self.lin4 = torch.nn.Linear(4*self.num_classes, self.num_classes)
    self.m = nn.Dropout(p=0.5)

  def forward(self, x, edge_index, batch, global_feat):
    x = F.relu(self.Conv1(x, edge_index))
    x = F.relu(self.Conv2(x, edge_index))
    x = F.relu(self.Conv3(x, edge_index))

    x = global_mean_pool(x, batch)
    # Global features are appended to the latent variables
    x = torch.cat((x, global_feat), dim=1)
    x = F.relu(self.lin1(x))
    x = self.m(x)
    x = F.relu(self.lin2(x))
    x = self.m(x)
    x = F.relu(self.lin3(x))
    x = F.sigmoid(self.lin4(x))
    return x

from torch_geometric.nn import SAGEConv

class SAGEClassifier(torch.nn.Module):
  def __init__(self, input_embed_dim : int, latent_dim = None,
num_classes : int = 2):
    super(SAGEClassifier, self).__init__()
    self.node_dim = input_embed_dim
    if latent_dim is None:
      self.latent_dim = self.node_dim
    else:
      self.latent_dim = latent_dim
    self.num_classes = num_classes
    self.Conv1 = SAGEConv(self.node_dim, self.latent_dim, project =
True)
    self.Conv2 = SAGEConv(self.latent_dim, 2*self.latent_dim, project
= True)
    self.Conv3 = SAGEConv(2*self.latent_dim, 4*self.latent_dim,
```

```
project = True)
    self.lin1 = torch.nn.Linear(4*self.latent_dim + 2,
2*self.latent_dim)
    self.lin2 = torch.nn.Linear(2*self.latent_dim, self.latent_dim)
    self.lin3 = torch.nn.Linear(self.latent_dim, 4*self.num_classes)
    self.lin4 = torch.nn.Linear(4*self.num_classes, self.num_classes)
    self.m = nn.Dropout(p=0.5)

  def forward(self, x, edge_index, batch, global_feat):
    x = self.Conv1(x, edge_index)
    x = self.Conv2(x, edge_index)
    x = self.Conv3(x, edge_index)

    x = global_mean_pool(x, batch)
    # Global features are appended to the latent variables
    x = torch.cat((x, global_feat), dim=1)
    x = F.relu(self.lin1(x))
    x = self.m(x)
    x = F.relu(self.lin2(x))
    x = self.m(x)
    x = F.relu(self.lin3(x))
    x = F.sigmoid(self.lin4(x))
    return x

from torch_geometric.nn import GATConv

class GATClassifier(torch.nn.Module):
  def __init__(self, input_embed_dim : int, latent_dim = None,
num_classes : int = 2, attn_heads = None):
    super(GATClassifier, self).__init__()
    self.node_dim = input_embed_dim
    if latent_dim is None:
      self.latent_dim = self.node_dim
    else:
      self.latent_dim = latent_dim
    if attn_heads is None:
      self.attn_heads = 3
    else:
      self.attn_heads = attn_heads
    self.num_classes = num_classes
    self.Conv1 = GATConv(self.node_dim, self.latent_dim)
    self.Conv2 = GATConv(self.latent_dim, 2*self.latent_dim)
    self.Conv3 = GATConv(2*self.latent_dim, 4*self.latent_dim)
    self.lin1 = torch.nn.Linear(4*self.latent_dim + 2,
2*self.latent_dim)
    self.lin2 = torch.nn.Linear(2*self.latent_dim, self.latent_dim)
    self.lin3 = torch.nn.Linear(self.latent_dim, 4*self.num_classes)
    self.lin4 = torch.nn.Linear(4*self.num_classes, self.num_classes)
    self.m = nn.Dropout(p=0.5)
```

```
  def forward(self, x, edge_index, batch, global_feat):
    x = F.relu(self.Conv1(x, edge_index))
    x = F.relu(self.Conv2(x, edge_index))
    x = F.relu(self.Conv3(x, edge_index))

    x = global_mean_pool(x, batch)
    # Global features are appended to the latent variables
    x = torch.cat((x, global_feat), dim=1)
    x = F.relu(self.lin1(x))
    x = self.m(x)
    x = F.relu(self.lin2(x))
    x = self.m(x)
    x = F.relu(self.lin3(x))
    x = F.sigmoid(self.lin4(x))
    return x

def train():
    model.train()

    for data in train_loader:
         data = data.to(torch.device('cuda'))
         out = model(data.x, data.edge_index, data.batch,
data.global_feat)
         loss = criterion(out, data.y)
         loss.backward()
         optimizer.step()
         optimizer.zero_grad()

def test(loader):
     model.eval()

     correct = 0
     for data in loader:
         data = data.to(torch.device('cuda'))
         out = model(data.x, data.edge_index, data.batch,
data.global_feat)
         pred = out.argmax(dim=1)
         correct += int((pred == data.y).sum())
     return correct / len(loader.dataset)

model = SAGEClassifier(5,32).to(torch.device('cuda'))
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
criterion = torch.nn.CrossEntropyLoss()

for epoch in range(60):
    train()
    train_acc = test(train_loader)
    test_acc = test(test_loader)
    val_acc = test(val_loader)
```

```
    print(f'Epoch: {epoch:03d}, Train Acc: {train_acc:.4f}, Test Acc:
{test_acc:.4f}, Val Acc: {val_acc:.4f}')
```

```
Epoch: 000, Train Acc: 0.4993, Test Acc: 0.5040, Val Acc: 0.4960
Epoch: 001, Train Acc: 0.4993, Test Acc: 0.5040, Val Acc: 0.4960
Epoch: 002, Train Acc: 0.4993, Test Acc: 0.5040, Val Acc: 0.4960
Epoch: 003, Train Acc: 0.5109, Test Acc: 0.5140, Val Acc: 0.5110
Epoch: 004, Train Acc: 0.5006, Test Acc: 0.4960, Val Acc: 0.5040
Epoch: 005, Train Acc: 0.6746, Test Acc: 0.6880, Val Acc: 0.6910
Epoch: 006, Train Acc: 0.6953, Test Acc: 0.7330, Val Acc: 0.7000
Epoch: 007, Train Acc: 0.6993, Test Acc: 0.7390, Val Acc: 0.6990
Epoch: 008, Train Acc: 0.6949, Test Acc: 0.7330, Val Acc: 0.6970
Epoch: 009, Train Acc: 0.6999, Test Acc: 0.7340, Val Acc: 0.7040
Epoch: 010, Train Acc: 0.6975, Test Acc: 0.7320, Val Acc: 0.7010
Epoch: 011, Train Acc: 0.7000, Test Acc: 0.7420, Val Acc: 0.6920
Epoch: 012, Train Acc: 0.6977, Test Acc: 0.7390, Val Acc: 0.6950
Epoch: 013, Train Acc: 0.7009, Test Acc: 0.7420, Val Acc: 0.6990
Epoch: 014, Train Acc: 0.7016, Test Acc: 0.7430, Val Acc: 0.7030
Epoch: 015, Train Acc: 0.7007, Test Acc: 0.7460, Val Acc: 0.6960
Epoch: 016, Train Acc: 0.6975, Test Acc: 0.7180, Val Acc: 0.7010
Epoch: 017, Train Acc: 0.7035, Test Acc: 0.7460, Val Acc: 0.7000
Epoch: 018, Train Acc: 0.7033, Test Acc: 0.7350, Val Acc: 0.7050
Epoch: 019, Train Acc: 0.7034, Test Acc: 0.7300, Val Acc: 0.7030
Epoch: 020, Train Acc: 0.7040, Test Acc: 0.7370, Val Acc: 0.7090
Epoch: 021, Train Acc: 0.7023, Test Acc: 0.7290, Val Acc: 0.7050
Epoch: 022, Train Acc: 0.7021, Test Acc: 0.7370, Val Acc: 0.7020
Epoch: 023, Train Acc: 0.7029, Test Acc: 0.7320, Val Acc: 0.6940
Epoch: 024, Train Acc: 0.6970, Test Acc: 0.7080, Val Acc: 0.7090
Epoch: 025, Train Acc: 0.7030, Test Acc: 0.7430, Val Acc: 0.6990
Epoch: 026, Train Acc: 0.7037, Test Acc: 0.7400, Val Acc: 0.7050
Epoch: 027, Train Acc: 0.7019, Test Acc: 0.7300, Val Acc: 0.7100
Epoch: 028, Train Acc: 0.6944, Test Acc: 0.6990, Val Acc: 0.7030
Epoch: 029, Train Acc: 0.6993, Test Acc: 0.7350, Val Acc: 0.6900
Epoch: 030, Train Acc: 0.7036, Test Acc: 0.7450, Val Acc: 0.6990
Epoch: 031, Train Acc: 0.7040, Test Acc: 0.7380, Val Acc: 0.7080
Epoch: 032, Train Acc: 0.7037, Test Acc: 0.7370, Val Acc: 0.7080
Epoch: 033, Train Acc: 0.7055, Test Acc: 0.7390, Val Acc: 0.7050
Epoch: 034, Train Acc: 0.7049, Test Acc: 0.7330, Val Acc: 0.7060
Epoch: 035, Train Acc: 0.7060, Test Acc: 0.7320, Val Acc: 0.7130
Epoch: 036, Train Acc: 0.7033, Test Acc: 0.7120, Val Acc: 0.7150
Epoch: 037, Train Acc: 0.7024, Test Acc: 0.7190, Val Acc: 0.7090
Epoch: 038, Train Acc: 0.6996, Test Acc: 0.7140, Val Acc: 0.7110
Epoch: 039, Train Acc: 0.7030, Test Acc: 0.7340, Val Acc: 0.6940
Epoch: 040, Train Acc: 0.7085, Test Acc: 0.7340, Val Acc: 0.7050
Epoch: 041, Train Acc: 0.7046, Test Acc: 0.7320, Val Acc: 0.7010
Epoch: 042, Train Acc: 0.7046, Test Acc: 0.7310, Val Acc: 0.7130
Epoch: 043, Train Acc: 0.7026, Test Acc: 0.7110, Val Acc: 0.7080
Epoch: 044, Train Acc: 0.7049, Test Acc: 0.7250, Val Acc: 0.7130
Epoch: 045, Train Acc: 0.7084, Test Acc: 0.7400, Val Acc: 0.7080
Epoch: 046, Train Acc: 0.7001, Test Acc: 0.7300, Val Acc: 0.6970
```

```
Epoch: 047, Train Acc: 0.7051, Test Acc: 0.7330, Val Acc: 0.6960
Epoch: 048, Train Acc: 0.7016, Test Acc: 0.7090, Val Acc: 0.7100
Epoch: 049, Train Acc: 0.7031, Test Acc: 0.7340, Val Acc: 0.6960
Epoch: 050, Train Acc: 0.7050, Test Acc: 0.7290, Val Acc: 0.7050
Epoch: 051, Train Acc: 0.7011, Test Acc: 0.7350, Val Acc: 0.6960
Epoch: 052, Train Acc: 0.7061, Test Acc: 0.7430, Val Acc: 0.7120
Epoch: 053, Train Acc: 0.7020, Test Acc: 0.7320, Val Acc: 0.6920
Epoch: 054, Train Acc: 0.7056, Test Acc: 0.7370, Val Acc: 0.7030
Epoch: 055, Train Acc: 0.7054, Test Acc: 0.7250, Val Acc: 0.7050
Epoch: 056, Train Acc: 0.7101, Test Acc: 0.7290, Val Acc: 0.7110
Epoch: 057, Train Acc: 0.7086, Test Acc: 0.7400, Val Acc: 0.6970
Epoch: 058, Train Acc: 0.7074, Test Acc: 0.7430, Val Acc: 0.7080
Epoch: 059, Train Acc: 0.7081, Test Acc: 0.7310, Val Acc: 0.6960
```

```
test(val_loader)
```

```
0.696
```

```
torch.save(model, "sage_gnn_gfc.pth")
```

```
model = GCNClassifier(5,32).to(torch.device('cuda'))
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
criterion = torch.nn.CrossEntropyLoss()

for epoch in range(60):
    train()
    train_acc = test(train_loader)
    test_acc = test(test_loader)
    val_acc = test(val_loader)
    print(f'Epoch: {epoch:03d}, Train Acc: {train_acc:.4f}, Test Acc:
{test_acc:.4f}, Val Acc: {val_acc:.4f}')
```

```
Epoch: 000, Train Acc: 0.5008, Test Acc: 0.4960, Val Acc: 0.5040
Epoch: 001, Train Acc: 0.4993, Test Acc: 0.5040, Val Acc: 0.4960
Epoch: 002, Train Acc: 0.6079, Test Acc: 0.6240, Val Acc: 0.6100
Epoch: 003, Train Acc: 0.5316, Test Acc: 0.5400, Val Acc: 0.5280
Epoch: 004, Train Acc: 0.6358, Test Acc: 0.6280, Val Acc: 0.6350
Epoch: 005, Train Acc: 0.6624, Test Acc: 0.6570, Val Acc: 0.6500
Epoch: 006, Train Acc: 0.6496, Test Acc: 0.6440, Val Acc: 0.6400
Epoch: 007, Train Acc: 0.6620, Test Acc: 0.6750, Val Acc: 0.6520
Epoch: 008, Train Acc: 0.6661, Test Acc: 0.6650, Val Acc: 0.6500
Epoch: 009, Train Acc: 0.6673, Test Acc: 0.6620, Val Acc: 0.6590
Epoch: 010, Train Acc: 0.6931, Test Acc: 0.7020, Val Acc: 0.6860
Epoch: 011, Train Acc: 0.7013, Test Acc: 0.7070, Val Acc: 0.6870
Epoch: 012, Train Acc: 0.7076, Test Acc: 0.7220, Val Acc: 0.7000
Epoch: 013, Train Acc: 0.7053, Test Acc: 0.7280, Val Acc: 0.6950
Epoch: 014, Train Acc: 0.7045, Test Acc: 0.7180, Val Acc: 0.7030
Epoch: 015, Train Acc: 0.6783, Test Acc: 0.6980, Val Acc: 0.6770
Epoch: 016, Train Acc: 0.7061, Test Acc: 0.7250, Val Acc: 0.7060
Epoch: 017, Train Acc: 0.7097, Test Acc: 0.7300, Val Acc: 0.7020
```

```
Epoch: 018, Train Acc: 0.7103, Test Acc: 0.7280, Val Acc: 0.7070
Epoch: 019, Train Acc: 0.7110, Test Acc: 0.7330, Val Acc: 0.7020
Epoch: 020, Train Acc: 0.7074, Test Acc: 0.7280, Val Acc: 0.6910
Epoch: 021, Train Acc: 0.7053, Test Acc: 0.7210, Val Acc: 0.7050
Epoch: 022, Train Acc: 0.7106, Test Acc: 0.7320, Val Acc: 0.7010
Epoch: 023, Train Acc: 0.7079, Test Acc: 0.7240, Val Acc: 0.7040
Epoch: 024, Train Acc: 0.7111, Test Acc: 0.7250, Val Acc: 0.7060
Epoch: 025, Train Acc: 0.7106, Test Acc: 0.7240, Val Acc: 0.7100
Epoch: 026, Train Acc: 0.7101, Test Acc: 0.7260, Val Acc: 0.7080
Epoch: 027, Train Acc: 0.7026, Test Acc: 0.7180, Val Acc: 0.7060
Epoch: 028, Train Acc: 0.7079, Test Acc: 0.7280, Val Acc: 0.7090
Epoch: 029, Train Acc: 0.7050, Test Acc: 0.7210, Val Acc: 0.7060
Epoch: 030, Train Acc: 0.7083, Test Acc: 0.7270, Val Acc: 0.6980
Epoch: 031, Train Acc: 0.7080, Test Acc: 0.7300, Val Acc: 0.7000
Epoch: 032, Train Acc: 0.7111, Test Acc: 0.7320, Val Acc: 0.6950
Epoch: 033, Train Acc: 0.7091, Test Acc: 0.7240, Val Acc: 0.7060
Epoch: 034, Train Acc: 0.6981, Test Acc: 0.7100, Val Acc: 0.7010
Epoch: 035, Train Acc: 0.7117, Test Acc: 0.7240, Val Acc: 0.7080
Epoch: 036, Train Acc: 0.7009, Test Acc: 0.7130, Val Acc: 0.7020
Epoch: 037, Train Acc: 0.7154, Test Acc: 0.7290, Val Acc: 0.7000
Epoch: 038, Train Acc: 0.7107, Test Acc: 0.7320, Val Acc: 0.7000
Epoch: 039, Train Acc: 0.7081, Test Acc: 0.7350, Val Acc: 0.7000
Epoch: 040, Train Acc: 0.7116, Test Acc: 0.7200, Val Acc: 0.7030
Epoch: 041, Train Acc: 0.7130, Test Acc: 0.7260, Val Acc: 0.7010
Epoch: 042, Train Acc: 0.7137, Test Acc: 0.7350, Val Acc: 0.7070
Epoch: 043, Train Acc: 0.7143, Test Acc: 0.7260, Val Acc: 0.7070
Epoch: 044, Train Acc: 0.7073, Test Acc: 0.7270, Val Acc: 0.7080
Epoch: 045, Train Acc: 0.7116, Test Acc: 0.7260, Val Acc: 0.7060
Epoch: 046, Train Acc: 0.7121, Test Acc: 0.7290, Val Acc: 0.6980
Epoch: 047, Train Acc: 0.7151, Test Acc: 0.7290, Val Acc: 0.7070
Epoch: 048, Train Acc: 0.7140, Test Acc: 0.7280, Val Acc: 0.7050
Epoch: 049, Train Acc: 0.7146, Test Acc: 0.7250, Val Acc: 0.6970
Epoch: 050, Train Acc: 0.7130, Test Acc: 0.7350, Val Acc: 0.6990
Epoch: 051, Train Acc: 0.7080, Test Acc: 0.7220, Val Acc: 0.7010
Epoch: 052, Train Acc: 0.7144, Test Acc: 0.7300, Val Acc: 0.6970
Epoch: 053, Train Acc: 0.7074, Test Acc: 0.7220, Val Acc: 0.7040
Epoch: 054, Train Acc: 0.7055, Test Acc: 0.7270, Val Acc: 0.6940
Epoch: 055, Train Acc: 0.7137, Test Acc: 0.7320, Val Acc: 0.6960
Epoch: 056, Train Acc: 0.7125, Test Acc: 0.7260, Val Acc: 0.7080
Epoch: 057, Train Acc: 0.7067, Test Acc: 0.7280, Val Acc: 0.6990
Epoch: 058, Train Acc: 0.7150, Test Acc: 0.7310, Val Acc: 0.6980
Epoch: 059, Train Acc: 0.7095, Test Acc: 0.7220, Val Acc: 0.7100
```

```
test(val_loader)
```

```
0.71
```

```
torch.save(model, "gcn_gnn_gfc.pth")
```

```
model = GATClassifier(5,32).to(torch.device('cuda'))
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
criterion = torch.nn.CrossEntropyLoss()

for epoch in range(60):
    train()
    train_acc = test(train_loader)
    test_acc = test(test_loader)
    val_acc = test(val_loader)
    print(f'Epoch: {epoch:03d}, Train Acc: {train_acc:.4f}, Test Acc:
{test_acc:.4f}, Val Acc: {val_acc:.4f}')
```

```
Epoch: 000, Train Acc: 0.5008, Test Acc: 0.4960, Val Acc: 0.5040
Epoch: 001, Train Acc: 0.5009, Test Acc: 0.5050, Val Acc: 0.4990
Epoch: 002, Train Acc: 0.5666, Test Acc: 0.5710, Val Acc: 0.5700
Epoch: 003, Train Acc: 0.6757, Test Acc: 0.6790, Val Acc: 0.6610
Epoch: 004, Train Acc: 0.6941, Test Acc: 0.7320, Val Acc: 0.7030
Epoch: 005, Train Acc: 0.6865, Test Acc: 0.7160, Val Acc: 0.6810
Epoch: 006, Train Acc: 0.6980, Test Acc: 0.7290, Val Acc: 0.7050
Epoch: 007, Train Acc: 0.6965, Test Acc: 0.7270, Val Acc: 0.7050
Epoch: 008, Train Acc: 0.7029, Test Acc: 0.7290, Val Acc: 0.7040
Epoch: 009, Train Acc: 0.7044, Test Acc: 0.7300, Val Acc: 0.7060
Epoch: 010, Train Acc: 0.7073, Test Acc: 0.7360, Val Acc: 0.7050
Epoch: 011, Train Acc: 0.7057, Test Acc: 0.7320, Val Acc: 0.7070
Epoch: 012, Train Acc: 0.7100, Test Acc: 0.7350, Val Acc: 0.7040
Epoch: 013, Train Acc: 0.7080, Test Acc: 0.7370, Val Acc: 0.7010
Epoch: 014, Train Acc: 0.7079, Test Acc: 0.7280, Val Acc: 0.7000
Epoch: 015, Train Acc: 0.7093, Test Acc: 0.7320, Val Acc: 0.7080
Epoch: 016, Train Acc: 0.7046, Test Acc: 0.7270, Val Acc: 0.6990
Epoch: 017, Train Acc: 0.7111, Test Acc: 0.7380, Val Acc: 0.7050
Epoch: 018, Train Acc: 0.7106, Test Acc: 0.7340, Val Acc: 0.7020
Epoch: 019, Train Acc: 0.7151, Test Acc: 0.7330, Val Acc: 0.7060
Epoch: 020, Train Acc: 0.7053, Test Acc: 0.7280, Val Acc: 0.7000
Epoch: 021, Train Acc: 0.7129, Test Acc: 0.7180, Val Acc: 0.7100
Epoch: 022, Train Acc: 0.7107, Test Acc: 0.7210, Val Acc: 0.7050
Epoch: 023, Train Acc: 0.7173, Test Acc: 0.7320, Val Acc: 0.7080
Epoch: 024, Train Acc: 0.7097, Test Acc: 0.7240, Val Acc: 0.7090
Epoch: 025, Train Acc: 0.7155, Test Acc: 0.7310, Val Acc: 0.7160
Epoch: 026, Train Acc: 0.7125, Test Acc: 0.7210, Val Acc: 0.7090
Epoch: 027, Train Acc: 0.7173, Test Acc: 0.7330, Val Acc: 0.7060
Epoch: 028, Train Acc: 0.7129, Test Acc: 0.7300, Val Acc: 0.7130
Epoch: 029, Train Acc: 0.7159, Test Acc: 0.7280, Val Acc: 0.7170
Epoch: 030, Train Acc: 0.7119, Test Acc: 0.7320, Val Acc: 0.7030
Epoch: 031, Train Acc: 0.7184, Test Acc: 0.7340, Val Acc: 0.7130
Epoch: 032, Train Acc: 0.7167, Test Acc: 0.7340, Val Acc: 0.7110
Epoch: 033, Train Acc: 0.7104, Test Acc: 0.7200, Val Acc: 0.7070
Epoch: 034, Train Acc: 0.7170, Test Acc: 0.7350, Val Acc: 0.7140
Epoch: 035, Train Acc: 0.7149, Test Acc: 0.7350, Val Acc: 0.7050
Epoch: 036, Train Acc: 0.7037, Test Acc: 0.7230, Val Acc: 0.7060
Epoch: 037, Train Acc: 0.7150, Test Acc: 0.7340, Val Acc: 0.7060
```

```
Epoch: 038, Train Acc: 0.7059, Test Acc: 0.7150, Val Acc: 0.7060
Epoch: 039, Train Acc: 0.7154, Test Acc: 0.7250, Val Acc: 0.7170
Epoch: 040, Train Acc: 0.7121, Test Acc: 0.7260, Val Acc: 0.7120
Epoch: 041, Train Acc: 0.7073, Test Acc: 0.7300, Val Acc: 0.7090
Epoch: 042, Train Acc: 0.7104, Test Acc: 0.7270, Val Acc: 0.7080
Epoch: 043, Train Acc: 0.7147, Test Acc: 0.7230, Val Acc: 0.7110
Epoch: 044, Train Acc: 0.7061, Test Acc: 0.7200, Val Acc: 0.7010
Epoch: 045, Train Acc: 0.7159, Test Acc: 0.7340, Val Acc: 0.7170
Epoch: 046, Train Acc: 0.7201, Test Acc: 0.7340, Val Acc: 0.7040
Epoch: 047, Train Acc: 0.7089, Test Acc: 0.7200, Val Acc: 0.7040
Epoch: 048, Train Acc: 0.7005, Test Acc: 0.7230, Val Acc: 0.6960
Epoch: 049, Train Acc: 0.7096, Test Acc: 0.7230, Val Acc: 0.7050
Epoch: 050, Train Acc: 0.7121, Test Acc: 0.7160, Val Acc: 0.7150
Epoch: 051, Train Acc: 0.7165, Test Acc: 0.7270, Val Acc: 0.7110
Epoch: 052, Train Acc: 0.7019, Test Acc: 0.7080, Val Acc: 0.7010
Epoch: 053, Train Acc: 0.7199, Test Acc: 0.7380, Val Acc: 0.7110
Epoch: 054, Train Acc: 0.7160, Test Acc: 0.7340, Val Acc: 0.7080
Epoch: 055, Train Acc: 0.7096, Test Acc: 0.7260, Val Acc: 0.7100
Epoch: 056, Train Acc: 0.7180, Test Acc: 0.7340, Val Acc: 0.7150
Epoch: 057, Train Acc: 0.7194, Test Acc: 0.7350, Val Acc: 0.7090
Epoch: 058, Train Acc: 0.7010, Test Acc: 0.7020, Val Acc: 0.7080
Epoch: 059, Train Acc: 0.7205, Test Acc: 0.7340, Val Acc: 0.7180
```

```
test(val_loader)
```

```
0.718
```

```
torch.save(model, "gat_gnn_gfc.pth")
```

```
del train_loader
del test_loader
del val_loader
```

Inference

1. The accuracy of each model, except SAGEConv(due to a sampling of neighbours giving Graph SAGE its efficiency), increased with higher k values.
2. The maximum recorded accuracy was 71.8% using the GAT-Classifier after 60 epochs.