```
!pip install --upgrade torch
!pip install torch-scatter torch-sparse torch-cluster torch-spline-conv torch-geometric -f https://data.pyg.org/whl/torch-1.11.0
!python -c "import torch; print(torch.__version__)"
!pip install torch pytorch-lightning
!pip install --upgrade pytorch-lightning
```

```
Requirement already satisfied: torch in /opt/conda/lib/python3.7/site-packages (1.9.1)
Collecting torch
  Downloading torch-1.11.0-cp37-cp37m-manylinux1_x86_64.whl (750.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 750.6/750.6 MB 1.4 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/site-packages (from torch) (4.1.1)
Installing collected packages: torch
  Attempting uninstall: torch
    Found existing installation: torch 1.9.1
    Uninstalling torch-1.9.1:
      Successfully uninstalled torch-1.9.1
Successfully installed torch-1.11.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package
Looking in links: https://data.pyg.org/whl/torch-1.11.0+cu102.html
Collecting torch-scatter
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu102/torch_scatter-2.0.9-cp37-cp37m-linux_x86_64.whl (8.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.0/8.0 MB 3.7 MB/s eta 0:00:0000:0100:010m
Collecting torch-sparse
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu102/torch_sparse-0.6.13-cp37-cp37m-linux_x86_64.whl (2.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 10.2 MB/s eta 0:00:00a 0:00:01
Collecting torch-cluster
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu102/torch_cluster-1.6.0-cp37-cp37m-linux_x86_64.whl (1.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 29.2 MB/s eta 0:00:00a 0:00:01
Collecting torch-spline-conv
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu102/torch_spline_conv-1.2.1-cp37-cp37m-linux_x86_64.whl (672 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 672.7/672.7 KB 12.6 MB/s eta 0:00:00a 0:00:01
Collecting torch-geometric
  Downloading torch_geometric-2.0.4.tar.gz (407 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 407.5/407.5 KB 1.4 MB/s eta 0:00:0000:0100:01
  Preparing metadata (setup.py) ... done
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from torch-sparse) (1.7.3)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (4.63.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (1.21.5)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (1.3.5)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (3.1.1)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (2.27.1)
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (3.0.7)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (from torch-geometric) (1.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-packages (from jinja2->torch-geometric) (2
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas->torch-geomet
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas->torch-geometric) (2021
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (from requests->torch-geometric) (3.
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packages (from requests->torch-geometri
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/lib/python3.7/site-packages (from requests->torch-g
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-packages (from requests->torch-geome
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn->torch-ge
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-packages (from scikit-learn->torch-geometric)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas->to
Building wheels for collected packages: torch-geometric
  Building wheel for torch-geometric (setup.py) ... done
  Created wheel for torch-geometric: filename=torch_geometric-2.0.4-py3-none-any.whl size=616603 sha256=724c8c927163827195
  Stored in directory: /root/.cache/pip/wheels/18/a6/a4/ca18c3051fcead866fe7b85700ee2240d883562a1bc70ce421
Successfully built torch-geometric
Installing collected packages: torch-spline-conv, torch-scatter, torch-cluster, torch-sparse, torch-geometric
Successfully installed torch-cluster-1.6.0 torch-geometric-2.0.4 torch-scatter-2.0.9 torch-sparse-0.6.13 torch-spline-conv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package
1.11.0+cu102
Requirement already satisfied: torch in /opt/conda/lib/python3.7/site-packages (1.11.0)
```

```
import numpy as np
import gc
import torch
import pyarrow as pa
from tqdm import tqdm
from pyarrow.parquet import ParquetFile
from sklearn.neighbors import kneighbors_graph
from sklearn.model_selection import train_test_split
from torch_geometric.data import Data
from torch_geometric.loader import DataLoader
```

```
pf = ParquetFile('../input/quarksgluons/QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272.test.snappy.parquet')
first_rows = next(pf.iter_batches(batch_size = 10000))
df = pa.Table.from_batches([first_rows]).to_pandas()
del first_rows
```

```
X_jets = np.array(np.array(np.array(df['X_jets'].tolist()).tolist()).tolist(), dtype='f')
X_jets = np.moveaxis(X_jets, 1, 3)
X_jets.shape
labels = torch.from_numpy(df['y'].to_numpy()).reshape(-1,1).type(torch.LongTensor)
print(labels.sum() *1.0/ len(labels))
del df
```

```
tensor(0.5038)
```

```
data = X_jets.reshape((-1,125*125,3))
non_black_pixels_mask = np.any(data != [0.,0.,0.], axis=-1)

node_list = []
for i, x in enumerate(data):
    node_list.append(x[non_black_pixels_mask[i]])
del X_jets
```

```
# from torch_geometric.utils import to_networkx
# import networkx as nx
# G = to_networkx(data, to_undirected=True)
# nx.draw(G)
```

```
dataset = []
for i,nodes in enumerate(tqdm(node_list)):
    edges = kneighbors_graph(nodes, 10, 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 = labels[i]
    dataset.append(Data(x=torch.from_numpy(nodes), edge_index=edge_list, edge_attr=edge_weight, y=y))
```

```
100%|██████████| 10000/10000 [00:45<00:00, 217.93it/s]
```

```
del labels, node_list, edge_list, edge_weight, y
gc.collect()
```

```
1199
```

```
data = dataset[0]
print(f'Number of nodes: {data.num_nodes}')
print(f'Number of edges: {data.num_edges}')
print(f'Number of node features: {data.num_node_features}')
print(f'Number of edges features: {data.num_edge_features}')
```

```
Number of nodes: 717
Number of edges: 7170
Number of node features: 3
Number of edges features: 1
```

```
rand_seed = 42
X_train, X_test = train_test_split(dataset, test_size=0.1, random_state = rand_seed)
X_train, X_val = train_test_split(X_train, test_size=0.1, random_state = rand_seed)
print(len(X_train), len(X_val), len(X_val))
```

```
8100 900 900
```

```
BATCH_SIZE = 64
train_loader = DataLoader(X_train, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(X_val, batch_size=BATCH_SIZE, shuffle=False)
test_loader = DataLoader(X_test, batch_size=BATCH_SIZE, shuffle=False)
batch = next(iter(test_loader))
print("Batch:", batch)
print("Labels:", batch.y[:10])
print("Batch indices:", batch.batch[:40])
```

```
Batch: DataBatch(x=[45127, 3], edge_index=[2, 451270], edge_attr=[451270, 1], y=[64], batch=[45127], ptr=[65])
Labels: tensor([0, 0, 1, 0, 0, 0, 0, 1, 0, 0])
Batch indices: tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
print(X_train[0])
print(X_train[1])
print(X_train[2])
```

```
Data(x=[713, 3], edge_index=[2, 7130], edge_attr=[7130, 1], y=[1])
Data(x=[928, 3], edge_index=[2, 9280], edge_attr=[9280, 1], y=[1])
Data(x=[784, 3], edge_index=[2, 7840], edge_attr=[7840, 1], y=[1])
```

https://colab.research.google.com/drive/1I8a0DfQ3fI7Njc62__mVXUlcAleUcInb?usp=sharing#scrollTo=0gZ-l0npPIca

```
## PyTorch
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.utils.data as data
import torch.optim as optim

import pytorch_lightning as pl
from pytorch_lightning.callbacks import LearningRateMonitor, ModelCheckpoint
```

```
from torch.nn import Linear
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, GraphConv, GATConv, TopKPooling
from torch_geometric.nn import global_mean_pool, global_max_pool

num_node_features = 3
num_classes = 2
num_head = 2

class GCN(torch.nn.Module):
    def __init__(self, c_in, c_hidden, c_out, dp_rate_linear=0.3):
        super().__init__()
        torch.manual_seed(123)
        self.conv1 = GATConv(c_in, c_hidden, heads=num_head)
        self.pool1 = TopKPooling(num_head*c_hidden, ratio=0.8)
        self.conv2 = GATConv(num_head*c_hidden, c_hidden, heads=num_head)
        self.pool2 = TopKPooling(num_head*c_hidden, ratio=0.8)
        self.conv3 = GATConv(num_head*c_hidden, c_hidden, heads=num_head)
        self.pool3 = TopKPooling(num_head*c_hidden, ratio=0.8)
        self.lin1 = Linear(num_head*c_hidden*2, c_hidden)
        self.lin2 = Linear(c_hidden, c_out)
        self.dp_rate_linear = dp_rate_linear

    def forward(self, x, edge_index, batch):
        x = self.conv1(x, edge_index)
        # x, edge_index, edge_attr, batch, perm, score[perm]
        x, edge_index, _, batch, _, _ = self.pool1(x, edge_index, None, batch)
        x1 = torch.cat([global_mean_pool(x, batch), global_max_pool(x, batch)], dim=1)
        x = x.relu()

        x = self.conv2(x, edge_index)
        x, edge_index, _, batch, _, _ = self.pool2(x, edge_index, None, batch)
        x2 = torch.cat([global_mean_pool(x, batch), global_max_pool(x, batch)], dim=1)
        x = x.relu()

        x = self.conv3(x, edge_index)
        x, edge_index, _, batch, _, _ = self.pool3(x, edge_index, None, batch)
        x3 = torch.cat([global_mean_pool(x, batch), global_max_pool(x, batch)], dim=1)

        x = x1 + x2 + x3

        # classifier
        x = F.dropout(x, p=self.dp_rate_linear, training=self.training)
        x = self.lin1(x)
        x = F.dropout(x, p=self.dp_rate_linear, training=self.training)
        x = self.lin2(x)

        return x

# model = GCN(hidden_channels=64)
# print(model)
```

```
from sklearn.metrics import roc_auc_score
learning_rate = 5e-4

class GraphLevelGNN(pl.LightningModule):

    def __init__(self, **model_kwargs):
        super().__init__()
        # Saving hyperparameters
        self.save_hyperparameters()

        self.model = GCN(**model_kwargs)
        self.loss_module = nn.BCEWithLogitsLoss() if self.hparams.c_out == 1 else nn.CrossEntropyLoss()

    def forward(self, data, mode="train"):
        x, edge_index, batch_idx = data.x, data.edge_index, data.batch
        # print(data.x.shape, data.edge_index.shape, data.batch.shape)

        x = self.model(x, edge_index, batch_idx)
        x = x.squeeze(dim=-1)

        if self.hparams.c_out == 1:
            preds = (x > 0).float()
            data.y = data.y.float()
        else:
            preds = x.argmax(dim=-1)
        loss = self.loss_module(x, data.y)
        acc = (preds == data.y).sum().float() / preds.shape[0]

        return loss, acc

    def configure_optimizers(self):
        optimizer = optim.Adam(self.parameters(), lr=learning_rate, weight_decay=0) # High lr because of small dataset and small m
        return optimizer

    def training_step(self, batch, batch_idx):
        loss, acc = self.forward(batch, mode="train")
        self.log('train_loss', loss, prog_bar=True)
        self.log('train_acc', acc, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        loss, acc = self.forward(batch, mode="val")
        self.log('val_loss', loss, prog_bar=True)
        self.log('val_acc', acc, prog_bar=True)

    def test_step(self, batch, batch_idx):
        loss, acc = self.forward(batch, mode="test")
        self.log('test_loss', loss, prog_bar=True)
        self.log('test_acc', acc, prog_bar=True)
```

```
CHECKPOINT_PATH = "./"
def train_graph_classifier(model_name, **model_kwargs):
    pl.seed_everything(42)

    # Create a PyTorch Lightning trainer with the generation callback
    root_dir = os.path.join(CHECKPOINT_PATH, "GraphLevel" + model_name)
    os.makedirs(root_dir, exist_ok=True)
    trainer = pl.Trainer(default_root_dir=root_dir,
                         callbacks=[ModelCheckpoint(save_weights_only=True, mode="max", monitor="val_acc")],
                         gpus=1 if str(device).startswith("cuda") else 0,
                         max_epochs=50,
                         progress_bar_refresh_rate=10)

    # Check whether pretrained model exists. If yes, load it and skip training
    model = GraphLevelGNN(**model_kwargs)
    print(model)
    trainer.fit(model, train_loader, val_loader)
    model = GraphLevelGNN.load_from_checkpoint(trainer.checkpoint_callback.best_model_path)

    # Test best model on validation and test set
    # train_result = trainer.test(model, dataloaders=train_loader, verbose=False)
    val_result = trainer.test(model, dataloaders=val_loader, verbose=False)
    test_result = trainer.test(model, dataloaders=test_loader, verbose=False)
    result = {"test": test_result[0]['test_acc'], "valid": val_result[0]['test_acc']}
    return model, result
```

```
import warnings

warnings.filterwarnings(
    "ignore", ".*Trying to infer the `batch_size` from an ambiguous collection.*"
)
```

```
import os
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
model, result = train_graph_classifier(model_name="GCN",c_in=3, c_hidden=32, c_out=2)
```

```
GraphLevelGNN(
  (model): GCN(
    (conv1): GATConv(3, 32, heads=2)
    (pool1): TopKPooling(64, ratio=0.8, multiplier=1.0)
    (conv2): GATConv(64, 32, heads=2)
    (pool2): TopKPooling(64, ratio=0.8, multiplier=1.0)
    (conv3): GATConv(64, 32, heads=2)
    (pool3): TopKPooling(64, ratio=0.8, multiplier=1.0)
    (lin1): Linear(in_features=128, out_features=32, bias=True)
    (lin2): Linear(in_features=32, out_features=2, bias=True)
  )
  (loss_module): CrossEntropyLoss()
)
Sanity Checking: 0it [00:00, ?it/s]
Training: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Validation: 0it [00:00, ?it/s]
Testing: 0it [00:00, ?it/s]
Testing: 0it [00:00, ?it/s]
```

```
print(f"Valid accuracy:  {100.0*result['valid']:4.2f}%")
print(f"Test accuracy:  {100.0*result['test']:4.2f}%")
```

```
Valid accuracy:  71.80%
Test accuracy:  71.59%
```