GNN_gsoc

March 21, 2023

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
!pip install pyg_lib torch_scatter torch_sparse torch_cluster torch_spline_conv␣
↪torch_geometric -f https://data.pyg.org/whl/torch-1.13.0+cu117.html
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Looking in links: https://data.pyg.org/whl/torch-1.13.0+cu117.html
Collecting pyg_lib
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu117/pyg_lib-
0.1.0%2Bpt113cu117-cp39-cp39-linux_x86_64.whl (1.9 MB)
                             1.9/1.9 MB
29.9 MB/s eta 0:00:00
Collecting torch_scatter
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu117/torch_scatter-
2.1.1%2Bpt113cu117-cp39-cp39-linux_x86_64.whl (10.1 MB)
                             10.1/10.1 MB
84.1 MB/s eta 0:00:00
Collecting torch_sparse
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu117/torch_sparse-
0.6.17%2Bpt113cu117-cp39-cp39-linux_x86_64.whl (4.7 MB)
                             4.7/4.7 MB
93.3 MB/s eta 0:00:00
Collecting torch_cluster
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu117/torch_cluster-
1.6.1%2Bpt113cu117-cp39-cp39-linux_x86_64.whl (3.2 MB)
                             3.2/3.2 MB
50.2 MB/s eta 0:00:00
Collecting torch_spline_conv
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu117/torch_spline_conv-
1.2.2%2Bpt113cu117-cp39-cp39-linux_x86_64.whl (872 kB)
                            872.2/872.2 KB
56.1 MB/s eta 0:00:00
Collecting torch_geometric
  Downloading torch_geometric-2.2.0.tar.gz (564 kB)
```

```
                                    565.0/565.0 KB
14.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages
(from torch_sparse) (1.10.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages
(from torch_geometric) (4.65.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages
(from torch_geometric) (1.22.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages
(from torch_geometric) (3.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from torch_geometric) (2.27.1)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.9/dist-
packages (from torch_geometric) (3.0.9)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-
packages (from torch_geometric) (1.2.2)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.9/dist-
packages (from torch_geometric) (5.9.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-
packages (from jinja2->torch_geometric) (2.1.2)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->torch_geometric)
(2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->torch_geometric) (3.4)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->torch_geometric) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->torch_geometric)
(1.26.15)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from scikit-learn->torch_geometric)
(3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-
packages (from scikit-learn->torch_geometric) (1.1.1)
Building wheels for collected packages: torch_geometric
  Building wheel for torch_geometric (setup.py) … done
  Created wheel for torch_geometric: filename=torch_geometric-2.2.0-py3-none-
any.whl size=773302
sha256=1272302adae652672c7f18ab98e7e48b62d9644ef7a2021e2341161d12d0fdbf
  Stored in directory: /root/.cache/pip/wheels/31/b2/8c/9b4bb72a4384eabd1ffeab2b
7ead692c9165e35711f8a9dc72
Successfully built torch_geometric
Installing collected packages: torch_spline_conv, torch_scatter, pyg_lib,
torch_sparse, torch_cluster, torch_geometric
Successfully installed pyg_lib-0.1.0+pt113cu117 torch_cluster-1.6.1+pt113cu117
torch_geometric-2.2.0 torch_scatter-2.1.1+pt113cu117
```

```
torch_sparse-0.6.17+pt113cu117 torch_spline_conv-1.2.2+pt113cu117
```

[3]:
```
import numpy as np
np.random.seed(0)
import os, glob
import time
import h5py
from tqdm import tqdm

import torch
import torch.nn.functional as F
import torch.optim as optim
import torch.utils
import torch.utils.data as data_utils
from torch.utils.data import ConcatDataset, Dataset, DataLoader, sampler,␣
 ↪DistributedSampler, Subset
from torch_geometric.data import Batch, Data, DataLoader
from torch_geometric.nn import GCNConv, global_mean_pool

from sklearn.metrics import roc_curve, auc
```

[4]:
```
BATCH = 8

granularity = 1
dropout = 0.3
maxnodes = 100
lr_init = 5.e-4
edgeconvblocks = 3
epochs = 5
os.environ["CUDA_VISIBLE_DEVICES"]=str(0)
PATH = "/content/drive/MyDrive/gsoc/quark-gluon_data-set_n139306.hdf5"
use_cuda = torch.cuda.is_available()
device = torch.device("cuda" if use_cuda else "cpu")
```

[5]:
```
import torch
from torch_geometric.data import Data
class HDF5Dataset(Dataset):
    def __init__(self, file_path):
        self.file = h5py.File(file_path, 'r')
        self.data = self.file['X_jets']
        self.m0 = self.file['m0']
        self.pt = self.file['pt']
        self.y = self.file['y']

    def __len__(self):
        return self.data.shape[0]
```

```
    def __getitem__(self, index):
        # Load X_jets in shape (125, 125, 3) from HDF5 file
        x_jets = self.data[index]
        # Transpose the array to shape (3, 125, 125)
        image = torch.tensor(x_jets.transpose(2, 0, 1)).detach().clone()

        nonzero_indices = torch.nonzero(image > 0)
        points = torch.zeros(nonzero_indices.shape[0], 3)
        points[:, 0] = 2.0 * (nonzero_indices[:, 2].float() / (image.shape[2] -␣
 ↪1) - 0.5) # x-coordinates
        points[:, 1] = 2.0 * (nonzero_indices[:, 1].float() / (image.shape[1] -␣
 ↪1) - 0.5) # y-coordinates
        points[:, 2] = image[nonzero_indices[:, 0], nonzero_indices[:, 1],␣
 ↪nonzero_indices[:, 2]]

        ret_data = {}
        ret_data["X_jets"] = torch.tensor(image)
        ret_data["points"] = torch.tensor(points)
        ret_data["m0"] = torch.tensor(self.m0[index])
        ret_data["pt"] = torch.tensor(self.pt[index])
        ret_data["y"] = torch.tensor(self.y[index])

        return dict(ret_data)
```

[23]:
```
dataset = HDF5Dataset(PATH)

indices = torch.arange(200)
train_dataset = Subset(dataset, indices)
```

[24]:
```
train_size = 0.8

val_dataset = train_dataset
test_dataset = train_dataset

num_train = len(train_dataset)
indices = list(range(num_train))
split = int(np.floor(train_size * num_train))
split2 = int(np.floor((train_size+(1-train_size)/2) * num_train))
np.random.shuffle(indices)
train_idx, valid_idx, test_idx = indices[:split], indices[split:split2],␣
 ↪indices[split2:]

train_data = Subset(train_dataset, indices=train_idx)
val_data = Subset(val_dataset, indices=valid_idx)
test_data = Subset(test_dataset, indices=test_idx)

train_data = Subset(train_dataset, indices=train_idx)
```

```
val_data = Subset(val_dataset, indices=valid_idx)
test_data = Subset(test_dataset, indices=test_idx)
```

[25]:
```
train_dataloader = DataLoader(train_data, shuffle=True, batch_size=BATCH,␣
 ↪num_workers=2)
val_dataloader = DataLoader(val_data, shuffle=True, batch_size=BATCH,␣
 ↪num_workers=2)
```

```
/usr/local/lib/python3.9/dist-packages/torch_geometric/deprecation.py:12:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

[26]:
```
class PointCloudGraphDataset(Dataset):
    def __init__(self, point_cloud_dataloader):
        self.point_cloud_dataloader = point_cloud_dataloader

    def __getitem__(self, index):
        points, label = self.point_cloud_dataloader.dataset[index]["points"],␣
 ↪self.point_cloud_dataloader.dataset[index]["y"]
        num_points = points.shape[0]
        edge_index = torch.zeros((2, num_points * (num_points - 1) // 2),␣
 ↪dtype=torch.long)
        edge_attr = torch.zeros(edge_index.shape[1], dtype=torch.float)
        k = 0
        for i in range(num_points):
            for j in range(i + 1, num_points):
                edge_index[0, k] = i
                edge_index[1, k] = j
                edge_attr[k] = torch.norm(points[i] - points[j])
                k += 1
        return Data(x=points, edge_index=edge_index, edge_attr=edge_attr,␣
 ↪y=label)

    def __len__(self):
        return len(self.point_cloud_dataloader.dataset)
```

[27]:
```
train_point_cloud_graph_dataset = PointCloudGraphDataset(train_dataloader)
val_point_cloud_graph_dataset = PointCloudGraphDataset(val_dataloader)
def collate_fn(batch):
    return Batch.from_data_list(batch)
train_dataloader = DataLoader(train_point_cloud_graph_dataset,␣
 ↪batch_size=BATCH, shuffle=True, collate_fn=collate_fn)
val_dataloader = DataLoader(val_point_cloud_graph_dataset, batch_size=BATCH,␣
 ↪shuffle=True, collate_fn=collate_fn)
```

[28]:
```
class GNNModel(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(GNNModel, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.conv3 = GCNConv(hidden_dim, hidden_dim)
        self.conv4 = GCNConv(hidden_dim, output_dim)

    def forward(self, data):
        x, edge_index, edge_attr = data.x, data.edge_index, data.edge_attr
        x = F.relu(self.conv1(x, edge_index, edge_attr))
        x = F.relu(self.conv2(x, edge_index, edge_attr))
        x = F.relu(self.conv3(x, edge_index, edge_attr))
        x = self.conv4(x, edge_index, edge_attr)
        x = global_mean_pool(x, data.batch)
        return F.log_softmax(x, dim=1)
```

[29]:
```
model = GNNModel(input_dim=3, hidden_dim=32, output_dim=2).to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
criterion = torch.nn.NLLLoss()

for epoch in range(50):
    model.train()
    for data in tqdm(train_dataloader):
        data = data.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, data.y.long())
        loss.backward()
        optimizer.step()

    model.eval()
    correct = 0
    total = 0
    for data in tqdm(val_dataloader):
        data = data.to(device)
        output = model(data)
        _, predicted = torch.max(output, 1)
        total += data.y.size(0)
        correct += (predicted == data.y).sum().item()
    accuracy = 100 * correct / total
    print('Epoch {}, Accuracy: {:.2f}%'.format(epoch + 1, accuracy))
```

```
  0%|          | 0/20 [00:00<?, ?it/s]<ipython-input-5-14a1bbc65325>:27:
UserWarning: To copy construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
```

```
torch.tensor(sourceTensor).
  ret_data["X_jets"] = torch.tensor(image)
<ipython-input-5-14a1bbc65325>:28: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  ret_data["points"] = torch.tensor(points)
100%|      | 20/20 [32:49<00:00, 98.46s/it]
100%|      | 3/3 [03:44<00:00, 74.86s/it]

Epoch 1, Accuracy: 50.00%

100%|      | 20/20 [33:36<00:00, 100.84s/it]
100%|      | 3/3 [03:51<00:00, 77.32s/it]

Epoch 2, Accuracy: 20.00%

100%|      | 20/20 [32:55<00:00, 98.78s/it]
100%|      | 3/3 [03:44<00:00, 74.76s/it]

Epoch 3, Accuracy: 75.00%

100%|      | 20/20 [32:55<00:00, 98.79s/it]
100%|      | 3/3 [03:51<00:00, 77.04s/it]

Epoch 4, Accuracy: 25.00%

100%|      | 20/20 [33:55<00:00, 101.79s/it]
100%|      | 3/3 [03:43<00:00, 74.48s/it]

Epoch 5, Accuracy: 55.00%

100%|      | 20/20 [32:56<00:00, 98.84s/it]
100%|      | 3/3 [03:40<00:00, 73.44s/it]

Epoch 6, Accuracy: 50.00%

  5%|          | 1/20 [01:37<30:53, 97.55s/it]
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-29-22c7211bbb35> in <module>
      5 for epoch in range(50):
      6     model.train()
----> 7     for data in tqdm(train_dataloader):
      8         data = data.to(device)
      9         optimizer.zero_grad()

/usr/local/lib/python3.9/dist-packages/tqdm/std.py in __iter__(self)
   1176
   1177         try:
-> 1178             for obj in iterable:
   1179                 yield obj
```

```
   1180                 # Update and possibly print the progressbar.

/usr/local/lib/python3.9/dist-packages/torch/utils/data/dataloader.py in␣
 ↪__next__(self)
    626                 # TODO(https://github.com/pytorch/pytorch/issues/76750)
    627                 self._reset()  # type: ignore[call-arg]
--> 628             data = self._next_data()
    629             self._num_yielded += 1
    630             if self._dataset_kind == _DatasetKind.Iterable and \

/usr/local/lib/python3.9/dist-packages/torch/utils/data/dataloader.py in␣
 ↪_next_data(self)
    669     def _next_data(self):
    670         index = self._next_index()  # may raise StopIteration
--> 671         data = self._dataset_fetcher.fetch(index)  # may raise␣
 ↪StopIteration
    672         if self._pin_memory:
    673             data = _utils.pin_memory.pin_memory(data, self.
 ↪_pin_memory_device)

/usr/local/lib/python3.9/dist-packages/torch/utils/data/_utils/fetch.py in␣
 ↪fetch(self, possibly_batched_index)
     56                 data = self.dataset.__getitems__(possibly_batched_index
     57             else:
---> 58                 data = [self.dataset[idx] for idx in␣
 ↪possibly_batched_index]
     59         else:
     60             data = self.dataset[possibly_batched_index]

/usr/local/lib/python3.9/dist-packages/torch/utils/data/_utils/fetch.py in␣
 ↪<listcomp>(.0)
     56                 data = self.dataset.__getitems__(possibly_batched_index
     57             else:
---> 58                 data = [self.dataset[idx] for idx in␣
 ↪possibly_batched_index]
     59         else:
     60             data = self.dataset[possibly_batched_index]

<ipython-input-26-8cec7e5252da> in __getitem__(self, index)
     12             for j in range(i + 1, num_points):
     13                 edge_index[0, k] = i
---> 14                 edge_index[1, k] = j
     15                 edge_attr[k] = torch.norm(points[i] - points[j])
     16                 k += 1

KeyboardInterrupt:
```