```
#checking pytorch version and making sure im using a gpu kernel
!python -c "import torch; print(torch.__version__)"
!python -c "import torch; print(torch.__version.cuda)"
!nvidia-smi
```

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'torch'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'torch'
Wed Jan  4 10:58:42 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   65C    P0    28W /  70W |    820MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Double-click (or enter) to edit

```
import torch

def format_pytorch_version(version):
  return version.split('+')[0]

TORCH_version = torch.__version__
TORCH = format_pytorch_version(TORCH_version)

def format_cuda_version(version):
  return 'cu' + version.replace('.', '')

CUDA_version = torch.version.cuda
CUDA = format_cuda_version(CUDA_version)

!pip install torch-scatter     -f https://pytorch-geometric.com/whl/torch-{TORCH}+{CUDA}.html
!pip install torch-sparse      -f https://pytorch-geometric.com/whl/torch-{TORCH}+{CUDA}.html
!pip install torch-cluster     -f https://pytorch-geometric.com/whl/torch-{TORCH}+{CUDA}.html
!pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-{TORCH}+{CUDA}.html
!pip install torch-geometric
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.13.0+cu116.html
Requirement already satisfied: torch-scatter in /usr/local/lib/python3.8/site-packages (2.1.0+pt113cu116)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.13.0+cu116.html
Requirement already satisfied: torch-sparse in /usr/local/lib/python3.8/site-packages (0.6.16+pt113cu116)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/site-packages (from torch-sparse) (1.10.0)
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in /usr/local/lib/python3.8/site-packages (from scipy->torch-sparse) (1.24.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.13.0+cu116.html
Requirement already satisfied: torch-cluster in /usr/local/lib/python3.8/site-packages (1.6.0+pt113cu116)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/site-packages (from torch-cluster) (1.10.0)
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in /usr/local/lib/python3.8/site-packages (from scipy->torch-cluster) (1.24.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.13.0+cu116.html
Requirement already satisfied: torch-spline-conv in /usr/local/lib/python3.8/site-packages (1.2.1+pt113cu116)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: torch-geometric in /usr/local/lib/python3.8/site-packages (2.2.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/site-packages (from torch-geometric) (1.24.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/site-packages (from torch-geometric) (1.10.0)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.8/site-packages (from torch-geometric) (5.9.4)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.8/site-packages (from torch-geometric) (3.0.9)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/site-packages (from torch-geometric) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.8/site-packages (from torch-geometric) (2.28.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/site-packages (from torch-geometric) (3.1.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/site-packages (from torch-geometric) (1.2.0)
```

```
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.8/site-packages (from jinja2->torch-geometric) (2.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/site-packages (from requests->torch-geometric) (1
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.8/site-packages (from requests->torch-geometric)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/site-packages (from requests->torch-geometric) (2022
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/site-packages (from requests->torch-geometric) (3.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/site-packages (from scikit-learn->torch-geometric)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/site-packages (from scikit-learn->torch-geometric) (1.2.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
```

```
!pip install -q condacolab
import condacolab
condacolab.install()
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
✨🍰✨ Everything looks OK!
```

```
!conda install -c rdkit rdkit
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

```
import rdkit
```

looking onto dataset

```
import rdkit
from torch_geometric.datasets import MoleculeNet
```

```
#loading ESOL/HIV dataset
data=MoleculeNet(root=",", name="HIV")
data
```

```
Downloading https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/HIV.csv
Processing...
Done!
HIV(41127)
```

```
# Investigating the dataset
print("Dataset type: ", type(data))
print("Dataset features: ", data.num_features)
print("Dataset target: ", data.num_classes)
print("Dataset length: ", data.len)
print("Dataset sample: ", data[0])
print("Sample  nodes: ", data[0].num_nodes)
print("Sample  edges: ", data[0].num_edges)
```

```
Dataset type:  <class 'torch_geometric.datasets.molecule_net.MoleculeNet'>
Dataset features:  9
Dataset target:  2
Dataset length:  <bound method InMemoryDataset.len of HIV(41127)>
Dataset sample:  Data(x=[19, 9], edge_index=[2, 40], edge_attr=[40, 3], smiles='CCC1=[O+][Cu-3]2([O+]=C(CC)C1)[O+]=C(CC)CC(CC)=[O+]
Sample  nodes:  19
Sample  edges:  40
```

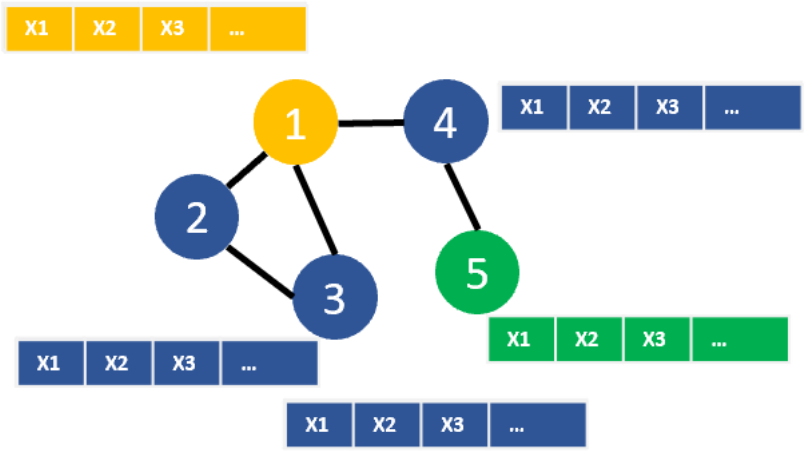
x vectors:node features y: target value/label down 1 node feature at first row, total 32 nodes for ESOL , so 32 feature vectors.

```
# Investigating the features
# Shape: [num_nodes, num_node_features]
data[0].x
```

```
tensor([[ 6,  0,  4,  5,  3,  0,  4,  0,  0],
        [ 6,  0,  4,  5,  2,  0,  4,  0,  0],
        [ 6,  0,  3,  5,  0,  0,  3,  0,  1],
        [ 8,  0,  2,  6,  0,  0,  3,  0,  1],
        [29,  0,  4,  2,  0,  0,  0,  0,  1],
        [ 8,  0,  2,  6,  0,  0,  3,  0,  1],
        [ 6,  0,  3,  5,  0,  0,  3,  0,  1],
        [ 6,  0,  4,  5,  2,  0,  4,  0,  0],
        [ 6,  0,  4,  5,  3,  0,  4,  0,  0],
```

```
[ 6,  0,  4,  5,  2,  0,  4,  0,  1],
[ 8,  0,  2,  6,  0,  0,  3,  0,  1],
[ 6,  0,  3,  5,  0,  0,  3,  0,  1],
[ 6,  0,  4,  5,  2,  0,  4,  0,  0],
[ 6,  0,  4,  5,  3,  0,  4,  0,  0],
[ 6,  0,  4,  5,  2,  0,  4,  0,  1],
[ 6,  0,  3,  5,  0,  0,  3,  0,  1],
[ 6,  0,  4,  5,  2,  0,  4,  0,  0],
[ 6,  0,  4,  5,  3,  0,  4,  0,  0],
[ 8,  0,  2,  6,  0,  0,  3,  0,  1]])
```

Double-click (or enter) to edit



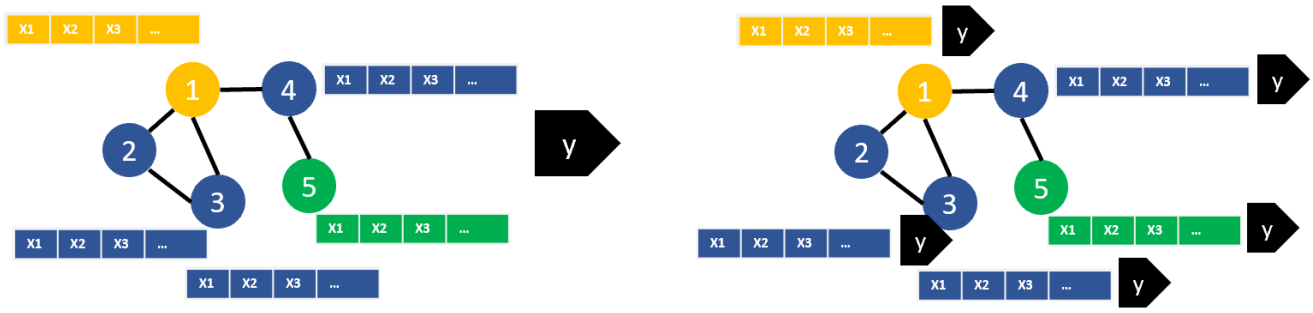

```
# Investigating the edges in sparse COO format
# Shape [2, num_edges]
data[0].edge_index.t()

# node 4 is connected to node 26
#node 5 is connected to 4
```

```
tensor([[ 0,  1],
        [ 1,  0],
        [ 1,  2],
        [ 2,  1],
        [ 2,  3],
        [ 2,  9],
        [ 3,  2],
        [ 3,  4],
        [ 4,  3],
        [ 4,  5],
        [ 4, 10],
        [ 4, 18],
        [ 5,  4],
        [ 5,  6],
        [ 6,  5],
        [ 6,  7],
        [ 6,  9],
        [ 7,  6],
        [ 7,  8],
        [ 8,  7],
        [ 9,  2],
        [ 9,  6],
        [10,  4],
        [10, 11],
        [11, 10],
        [11, 12],
        [11, 14],
        [12, 11],
        [12, 13],
        [13, 12],
        [14, 11],
        [14, 15],
        [15, 14],
        [15, 16],
        [15, 18],
        [16, 15],
        [16, 17],
        [17, 16],
        [18,  4],
        [18, 15]])
```

```
data[0].y
```

```
tensor([[0.]])
```

Double-click (or enter) to edit

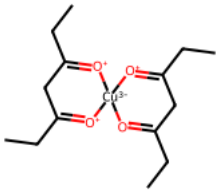


next, we want to have our SMILES molecules as graph SMILES to GRAPHS

```
data[0]["smiles"]
```

```
'CCC1=[O+][Cu-3]2([O+]=C(CC)C1)[O+]=C(CC)CC(CC)=[O+]2'
```

```
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
molecule = Chem.MolFromSmiles(data[0]["smiles"])
molecule
```



implementing GNN We will use different layers.....these layers come from torchgeometric.nn eg:GNN conv layer-simple message passing layer

In init():we will define our message passing layers

1. transformation layer: transform 9 node feature to an embedding vector of size:64
2. output layer: telling that we need to perform regression

forward(): we pass node features, edge info to this fn

different learning problems (node, edge or graph prediction) require different GNN architectures.

For example for node-level prediction will often encounter masks. For graph-level predictions on the other hand you need to combine the node embeddings.

```
import torch
from torch.nn import Linear
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, TopKPooling, global_mean_pool
from torch_geometric.nn import global_mean_pool as gap, global_max_pool as gmp
embedding_size = 64

class GCN(torch.nn.Module):
    def __init__(self):
        # Init parent
        super(GCN, self).__init__()
        torch.manual_seed(42)

        # GCN layers
        self.initial_conv = GCNConv(data.num_features, embedding_size)
        self.conv1 = GCNConv(embedding_size, embedding_size)
        self.conv2 = GCNConv(embedding_size, embedding_size)
        self.conv3 = GCNConv(embedding_size, embedding_size)
```

```
        # Output layer
        self.out = Linear(embedding_size*2, 1)

    def forward(self, x, edge_index, batch_index):
        # First Conv layer
        hidden = self.initial_conv(x, edge_index)#message passing
        hidden = F.tanh(hidden)#activation fn

        # Other Conv layers
        hidden = self.conv1(hidden, edge_index)
        hidden = F.tanh(hidden)
        hidden = self.conv2(hidden, edge_index)
        hidden = F.tanh(hidden)
        hidden = self.conv3(hidden, edge_index)
        hidden = F.tanh(hidden)

        # Global Pooling (stack different aggregations)
        #we need to combine node states of 1 graph into a single representation.
        #for that here we use mean(mean of all the nodes) nd max pooling
        #concatenate the results of mean and max pooling


        hidden = torch.cat([gmp(hidden, batch_index),
                            gap(hidden, batch_index)], dim=1)

        # Apply a final (linear) classifier.ie, that single vector rep all nodes
#
#passing to final o/p layer
        out = self.out(hidden)

        return out, hidden

model = GCN()
print(model)
print("Number of parameters: ", sum(p.numel() for p in model.parameters()))
```

```
GCN(
  (initial_conv): GCNConv(9, 64)
  (conv1): GCNConv(64, 64)
  (conv2): GCNConv(64, 64)
  (conv3): GCNConv(64, 64)
  (out): Linear(in_features=128, out_features=1, bias=True)
)
Number of parameters:  13249
```

training the GNN

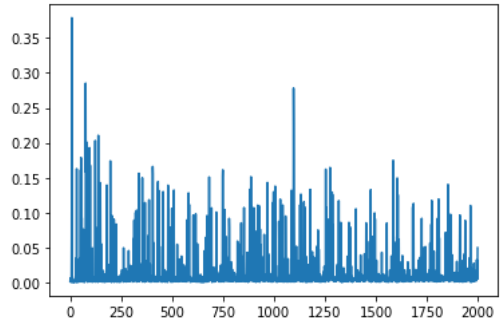
```
from torch_geometric.data import DataLoader
import warnings
warnings.filterwarnings("ignore")

# Root mean squared error
loss_fn = torch.nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.0007)

# Use GPU for training
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Wrap data in a data loader
data_size = len(data)
NUM_GRAPHS_PER_BATCH = 64
#64 molecules in each of these batches

loader = DataLoader(data[:int(data_size * 0.8)],
                    batch_size=NUM_GRAPHS_PER_BATCH, shuffle=True)
test_loader = DataLoader(data[int(data_size * 0.8):],
                         batch_size=NUM_GRAPHS_PER_BATCH, shuffle=True)
     #Dataloader to train data

def train(data):
    # Enumerate over the data
    for batch in loader:
      # Use GPU
      batch.to(device)
      # Reset gradients
      optimizer.zero_grad()
      # Passing the node features and the connection info
      pred, embedding = model(batch.x.float(), batch.edge_index, batch.batch)
      # Calculating the loss and gradients
      loss = loss_fn(pred, batch.y)
      loss.backward()
```

```
        # Update using the gradients
        optimizer.step()
    return loss, embedding

print("Starting training...")
losses = []
for epoch in range(2000):
    loss, h = train(data)
    losses.append(loss)
    if epoch % 100 == 0:
      print(f"Epoch {epoch} | Train Loss {loss}")
```

```
Starting training...
Epoch 0 | Train Loss 0.001605216064490378
Epoch 100 | Train Loss 0.007540540304034948
Epoch 200 | Train Loss 0.002204520395025611
Epoch 300 | Train Loss 0.003122455207630992
Epoch 400 | Train Loss 0.0022131141740828753
Epoch 500 | Train Loss 0.004110430832952261
Epoch 600 | Train Loss 0.007362925913184881
Epoch 700 | Train Loss 0.007860051468014717
Epoch 800 | Train Loss 0.007502086679714918
Epoch 900 | Train Loss 0.0040099106131494045
Epoch 1000 | Train Loss 0.01238179486244917
Epoch 1100 | Train Loss 0.0039007817395031452
Epoch 1300 | Train Loss 0.0013810101663693786
Epoch 1400 | Train Loss 0.004591164644807577
Epoch 1500 | Train Loss 0.005568958818912506
Epoch 1600 | Train Loss 0.01845824532210827
Epoch 1700 | Train Loss 0.003071602899581194
Epoch 1800 | Train Loss 0.005969869438558817
Epoch 1900 | Train Loss 0.0026812911964952946
```

```
# Visualize learning (training loss)# error is reducing
import seaborn as sns
losses_float = [float(loss.cpu().detach().numpy()) for loss in losses]
loss_indices = [i for i,l in enumerate(losses_float)]
plt = sns.lineplot(loss_indices, losses_float)
plt
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed8e0f1f40>
```



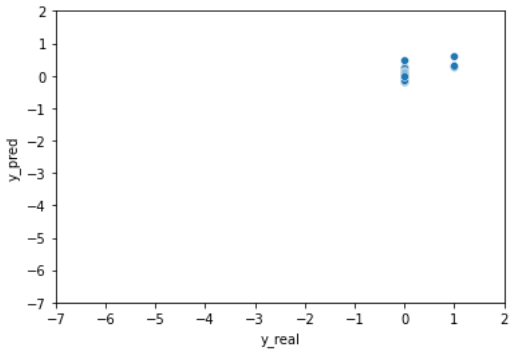
getting a test prediction

```
import pandas as pd

# Analyze the results for one batch
test_batch = next(iter(test_loader))
with torch.no_grad():
    test_batch.to(device)
    pred, embed = model(test_batch.x.float(), test_batch.edge_index, test_batch.batch)
    df = pd.DataFrame()
    df["y_real"] = test_batch.y.tolist()
    df["y_pred"] = pred.tolist()
df["y_real"] = df["y_real"].apply(lambda row: row[0])
df["y_pred"] = df["y_pred"].apply(lambda row: row[0])
df
```

	y_real	y_pred
0	0.0	0.068242
1	0.0	-0.115410
2	0.0	0.056601
3	0.0	-0.062921
4	0.0	-0.052742

```
plt = sns.scatterplot(data=df, x="y_real", y="y_pred")
plt.set(xlim=(-7, 2))
plt.set(ylim=(-7, 2))
plt
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed8e0f5070>
```



✓ 0s completed at 7:24 PM