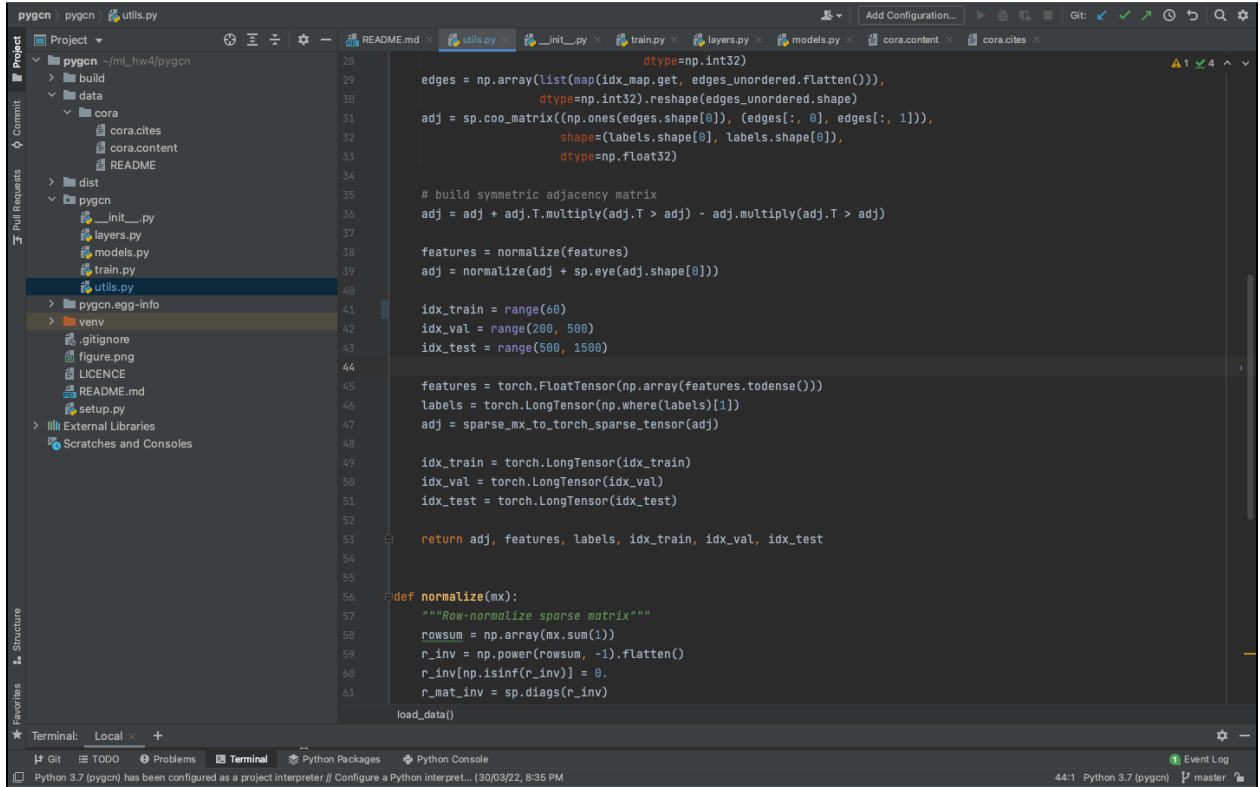


## Problem 4:

idx\_train=60



```
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61

def load_data():
    dtype=np.int32
    edges = np.array(list(map(idx_map.get, edges_unordered.flatten()))),
    dtype=np.int32).reshape(edges_unordered.shape)
    adj = sp.coo_matrix((np.ones(edges.shape[0]), (edges[:, 0], edges[:, 1])),
    shape=(Labels.shape[0], Labels.shape[0]),
    dtype=np.float32)

    # build symmetric adjacency matrix
    adj = adj + adj.T.multiply(adj.T > adj) - adj.multiply(adj.T > adj)

    features = normalize(features)
    adj = normalize(adj + sp.eye(adj.shape[0]))

    idx_train = range(60)
    idx_val = range(200, 500)
    idx_test = range(500, 1500)

    features = torch.FloatTensor(np.array(features.todense()))
    labels = torch.LongTensor(np.where(labels)[1])
    adj = sparse_mx_to_torch_sparse_tensor(adj)

    idx_train = torch.LongTensor(idx_train)
    idx_val = torch.LongTensor(idx_val)
    idx_test = torch.LongTensor(idx_test)

    return adj, features, labels, idx_train, idx_val, idx_test

def normalize(mx):
    """Row-normalize sparse matrix"""
    rowsum = np.array(mx.sum(1))
    r_inv = np.power(rowsum, -1).flatten()
    r_inv[np.isinf(r_inv)] = 0.
    r_mat_inv = sp.diags(r_inv)
    load_data()
```

Output:

```
pygcn > pygcn > utils.py
Project > Project > README.md > utils.py > _init_.py > train.py > layers.py > models.py > cora.content > cora.cites >
Terminal: Local > +
Epoch: 0170 loss_train: 0.3026 acc_train: 0.9500 loss_val: 0.9157 acc_val: 0.7233 time: 0.0171s
Epoch: 0171 loss_train: 0.3293 acc_train: 0.9500 loss_val: 0.9159 acc_val: 0.7200 time: 0.0206s
Epoch: 0172 loss_train: 0.3300 acc_train: 1.0000 loss_val: 0.9145 acc_val: 0.7233 time: 0.0264s
Epoch: 0173 loss_train: 0.2790 acc_train: 0.9667 loss_val: 0.9114 acc_val: 0.7167 time: 0.0247s
Epoch: 0174 loss_train: 0.3068 acc_train: 0.9833 loss_val: 0.9069 acc_val: 0.7200 time: 0.0237s
Epoch: 0175 loss_train: 0.2964 acc_train: 0.9667 loss_val: 0.9021 acc_val: 0.7200 time: 0.0268s
Epoch: 0176 loss_train: 0.3437 acc_train: 0.9333 loss_val: 0.8977 acc_val: 0.7200 time: 0.0236s
Epoch: 0177 loss_train: 0.3325 acc_train: 0.9500 loss_val: 0.8942 acc_val: 0.7200 time: 0.0227s
Epoch: 0178 loss_train: 0.2817 acc_train: 0.9667 loss_val: 0.8920 acc_val: 0.7200 time: 0.0193s
Epoch: 0179 loss_train: 0.2624 acc_train: 0.9833 loss_val: 0.8914 acc_val: 0.7167 time: 0.0210s
Epoch: 0180 loss_train: 0.2632 acc_train: 0.9667 loss_val: 0.8928 acc_val: 0.7167 time: 0.0213s
Epoch: 0181 loss_train: 0.3153 acc_train: 0.9667 loss_val: 0.8954 acc_val: 0.7167 time: 0.0241s
Epoch: 0182 loss_train: 0.2982 acc_train: 0.9667 loss_val: 0.8985 acc_val: 0.7167 time: 0.0157s
Epoch: 0183 loss_train: 0.2833 acc_train: 0.9500 loss_val: 0.9006 acc_val: 0.7167 time: 0.0194s
Epoch: 0184 loss_train: 0.2713 acc_train: 0.9833 loss_val: 0.9037 acc_val: 0.7200 time: 0.0216s
Epoch: 0185 loss_train: 0.2350 acc_train: 1.0000 loss_val: 0.9038 acc_val: 0.7200 time: 0.0234s
Epoch: 0186 loss_train: 0.3169 acc_train: 0.9333 loss_val: 0.9016 acc_val: 0.7200 time: 0.0222s
Epoch: 0187 loss_train: 0.3060 acc_train: 0.9833 loss_val: 0.8962 acc_val: 0.7167 time: 0.0199s
Epoch: 0188 loss_train: 0.2507 acc_train: 0.9667 loss_val: 0.8912 acc_val: 0.7233 time: 0.0194s
Epoch: 0189 loss_train: 0.2829 acc_train: 0.9833 loss_val: 0.8865 acc_val: 0.7233 time: 0.0243s
Epoch: 0190 loss_train: 0.2340 acc_train: 0.9667 loss_val: 0.8830 acc_val: 0.7300 time: 0.0211s
Epoch: 0191 loss_train: 0.2834 acc_train: 0.9833 loss_val: 0.8815 acc_val: 0.7333 time: 0.0205s
Epoch: 0192 loss_train: 0.2737 acc_train: 1.0000 loss_val: 0.8809 acc_val: 0.7300 time: 0.0203s
Epoch: 0193 loss_train: 0.2262 acc_train: 0.9667 loss_val: 0.8806 acc_val: 0.7300 time: 0.0220s
Epoch: 0194 loss_train: 0.2794 acc_train: 0.9667 loss_val: 0.8795 acc_val: 0.7233 time: 0.0240s
Epoch: 0195 loss_train: 0.2793 acc_train: 0.9833 loss_val: 0.8784 acc_val: 0.7300 time: 0.0161s
Epoch: 0196 loss_train: 0.2634 acc_train: 0.9667 loss_val: 0.8781 acc_val: 0.7300 time: 0.0185s
Epoch: 0197 loss_train: 0.2582 acc_train: 0.9667 loss_val: 0.8771 acc_val: 0.7300 time: 0.0229s
Epoch: 0198 loss_train: 0.2509 acc_train: 0.9667 loss_val: 0.8738 acc_val: 0.7300 time: 0.0233s
Epoch: 0199 loss_train: 0.2808 acc_train: 0.9667 loss_val: 0.8734 acc_val: 0.7300 time: 0.0187s
Epoch: 0200 loss_train: 0.3357 acc_train: 0.9833 loss_val: 0.8733 acc_val: 0.7300 time: 0.0246s
Optimization Finished!
Total time elapsed: 4.3976s
Test set results: loss= 0.8488 accuracy= 0.7320
(venv) (base) rahulmaddula@rahuls-Air pygcn %
```

idx\_train=120

```
pygcn > pygcn > utils.py
Project > Project > README.md > utils.py > _init_.py > train.py > layers.py > models.py > cora.content > cora.cites >
Terminal: Local > +
30 dtype=np.int32).reshape(edges_unordered.shape)
31 adj = sp.coo_matrix((np.ones(edges.shape[0]), (edges[:, 0], edges[:, 1])),
32 shape=(Labels.shape[0], Labels.shape[0]),
33 dtype=np.float32)
34
35 # build symmetric adjacency matrix
36 adj = adj + adj.T.multiply(adj.T>adj) - adj.multiply(adj.T>adj)
37
38 features = normalize(features)
39 adj = normalize(adj + sp.eye(adj.shape[0]))
40
41 idx_train = range(120)
42 idx_val = range(200, 500)
43 idx_test = range(500, 1500)
44
45 features = torch.FloatTensor(np.array(features.todense()))
46 labels = torch.LongTensor(np.where(labels)[1])
47 adj = sparse_mx_to_torch_sparse_tensor(adj)
48
49 idx_train = torch.LongTensor(idx_train)
50 idx_val = torch.LongTensor(idx_val)
51 idx_test = torch.LongTensor(idx_test)
52
53 return adj, features, labels, idx_train, idx_val, idx_test
54
55
56 def normalize(mx):
57     """Row-normalize sparse matrix"""
58     rowsum = np.array(mx.sum(1))
59     r_inv = np.power(rowsum, -1).flatten()
60     r_inv[np.isinf(r_inv)] = 0.
61     r_mat_inv = sp.diags(r_inv)
62     mx = r_mat_inv.dot(mx)
63     return mx
64 load_data()
```

Output:

```
pygcn | pygcn | utils.py
Project | README.md | utils.py | __init__.py | train.py | layers.py | models.py | cora.content | cora.cites
Terminal: Local +
Epoch: 0170 loss_train: 0.4798 acc_train: 0.9167 loss_val: 0.7610 acc_val: 0.8000 time: 0.0223s
Epoch: 0171 loss_train: 0.4611 acc_train: 0.9250 loss_val: 0.7604 acc_val: 0.8000 time: 0.0286s
Epoch: 0172 loss_train: 0.4593 acc_train: 0.9250 loss_val: 0.7600 acc_val: 0.8000 time: 0.0296s
Epoch: 0173 loss_train: 0.4639 acc_train: 0.9417 loss_val: 0.7610 acc_val: 0.7867 time: 0.0240s
Epoch: 0174 loss_train: 0.4423 acc_train: 0.9167 loss_val: 0.7588 acc_val: 0.7867 time: 0.0221s
Epoch: 0175 loss_train: 0.4016 acc_train: 0.9250 loss_val: 0.7555 acc_val: 0.7867 time: 0.0222s
Epoch: 0176 loss_train: 0.4393 acc_train: 0.9250 loss_val: 0.7527 acc_val: 0.7933 time: 0.0214s
Epoch: 0177 loss_train: 0.4302 acc_train: 0.9333 loss_val: 0.7495 acc_val: 0.7967 time: 0.0206s
Epoch: 0178 loss_train: 0.4352 acc_train: 0.9333 loss_val: 0.7477 acc_val: 0.7967 time: 0.0215s
Epoch: 0179 loss_train: 0.4817 acc_train: 0.9167 loss_val: 0.7458 acc_val: 0.7967 time: 0.0270s
Epoch: 0180 loss_train: 0.3843 acc_train: 0.9250 loss_val: 0.7443 acc_val: 0.7967 time: 0.0218s
Epoch: 0181 loss_train: 0.4131 acc_train: 0.9333 loss_val: 0.7419 acc_val: 0.7967 time: 0.0245s
Epoch: 0182 loss_train: 0.4193 acc_train: 0.9250 loss_val: 0.7398 acc_val: 0.7900 time: 0.0212s
Epoch: 0183 loss_train: 0.4402 acc_train: 0.9250 loss_val: 0.7394 acc_val: 0.7933 time: 0.0284s
Epoch: 0184 loss_train: 0.4345 acc_train: 0.9083 loss_val: 0.7390 acc_val: 0.8000 time: 0.0259s
Epoch: 0185 loss_train: 0.4051 acc_train: 0.9333 loss_val: 0.7386 acc_val: 0.8000 time: 0.0223s
Epoch: 0186 loss_train: 0.4609 acc_train: 0.9333 loss_val: 0.7358 acc_val: 0.7967 time: 0.0224s
Epoch: 0187 loss_train: 0.4088 acc_train: 0.9083 loss_val: 0.7326 acc_val: 0.7933 time: 0.0209s
Epoch: 0188 loss_train: 0.4380 acc_train: 0.9250 loss_val: 0.7322 acc_val: 0.7933 time: 0.0209s
Epoch: 0189 loss_train: 0.4440 acc_train: 0.8917 loss_val: 0.7314 acc_val: 0.7933 time: 0.0223s
Epoch: 0190 loss_train: 0.4116 acc_train: 0.9333 loss_val: 0.7310 acc_val: 0.7933 time: 0.0188s
Epoch: 0191 loss_train: 0.4353 acc_train: 0.9417 loss_val: 0.7308 acc_val: 0.7933 time: 0.0226s
Epoch: 0192 loss_train: 0.4450 acc_train: 0.9333 loss_val: 0.7299 acc_val: 0.7967 time: 0.0220s
Epoch: 0193 loss_train: 0.4201 acc_train: 0.9417 loss_val: 0.7286 acc_val: 0.7933 time: 0.0221s
Epoch: 0194 loss_train: 0.4039 acc_train: 0.9167 loss_val: 0.7268 acc_val: 0.7933 time: 0.0223s
Epoch: 0195 loss_train: 0.4152 acc_train: 0.9583 loss_val: 0.7249 acc_val: 0.7967 time: 0.0240s
Epoch: 0196 loss_train: 0.3818 acc_train: 0.9750 loss_val: 0.7237 acc_val: 0.7933 time: 0.0224s
Epoch: 0197 loss_train: 0.3891 acc_train: 0.9750 loss_val: 0.7230 acc_val: 0.8000 time: 0.0218s
Epoch: 0198 loss_train: 0.3998 acc_train: 0.9250 loss_val: 0.7233 acc_val: 0.8000 time: 0.0228s
Epoch: 0199 loss_train: 0.4118 acc_train: 0.9167 loss_val: 0.7237 acc_val: 0.8033 time: 0.0219s
Epoch: 0200 loss_train: 0.4390 acc_train: 0.9167 loss_val: 0.7241 acc_val: 0.8033 time: 0.0228s
Optimization Finished!
Total time elapsed: 4.7341s
Test set results: loss= 0.7656 accuracy= 0.8010
* (venv) (base) rahulmaddula@rahuls-Air pygcn %
```

idx\_train=180

```
pygcn | pygcn | utils.py
Project | README.md | utils.py | __init__.py | train.py | layers.py | models.py | cora.content | cora.cites
Terminal: Local +
34 # build symmetric adjacency matrix
35 adj = adj + adj.T.multiply(adj.T > adj) - adj.multiply(adj.T > adj)
36
37 features = normalize(features)
38 adj = normalize(adj + sp.eye(adj.shape[0]))
39
40
41 idx_train = range(180)
42 idx_val = range(200, 500)
43 idx_test = range(500, 1500)
44
45 features = torch.FloatTensor(np.array(features.todense()))
46 labels = torch.LongTensor(np.where(labels)[1])
47 adj = sparse_mx_to_torch_sparse_tensor(adj)
48
49 idx_train = torch.LongTensor(idx_train)
50 idx_val = torch.LongTensor(idx_val)
51 idx_test = torch.LongTensor(idx_test)
52
53 return adj, features, labels, idx_train, idx_val, idx_test
54
55
56 def normalize(mx):
57     """Row-normalize sparse matrix"""
58     rowsum = np.array(mx.sum(1))
59     r_inv = np.power(rowsum, -1).flatten()
60     r_inv[np.isinf(r_inv)] = 0.
61     r_mat_inv = sp.diags(r_inv)
62     mx = r_mat_inv.dot(mx)
63     return mx
64
65
66 def accuracy(output, labels):
67     preds = output.max(1)[1].type_as(labels)
68     load_data()
```

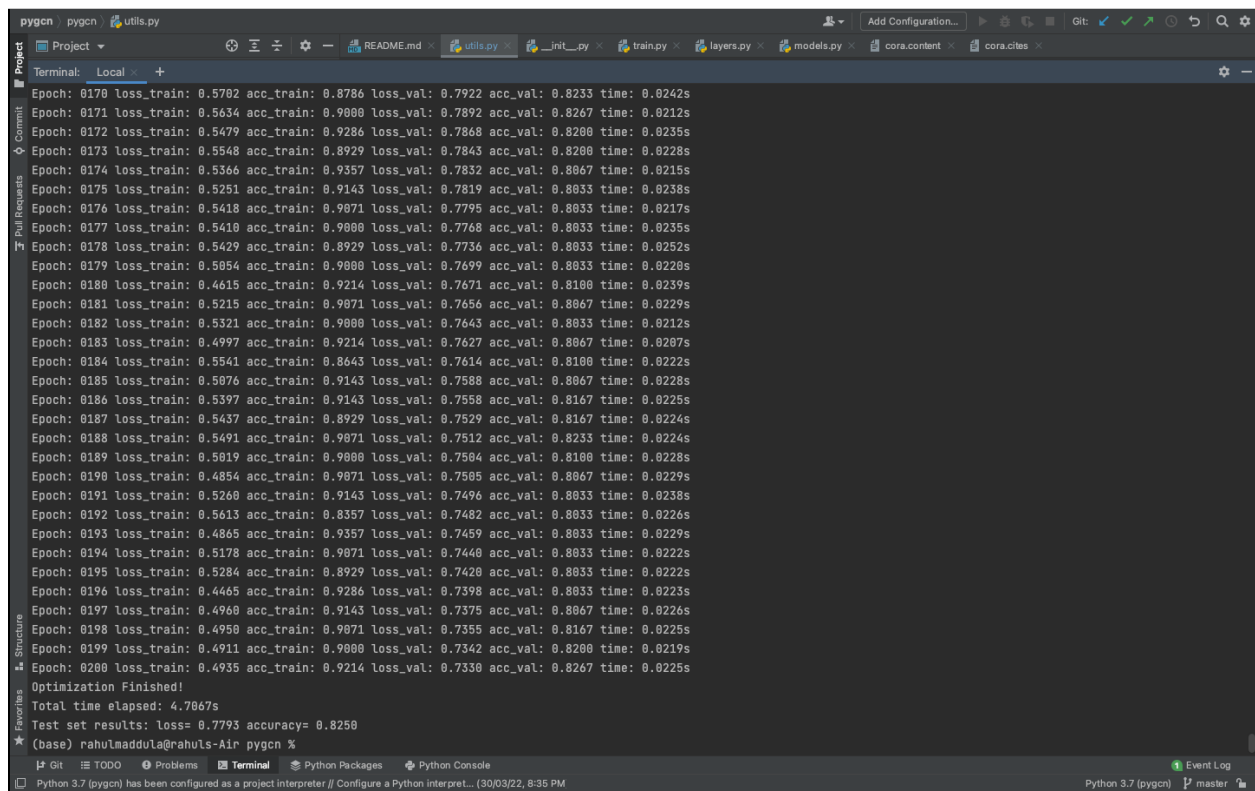
Output:

```
pygcn  pygcn  utils.py
Project
Terminal: Local +
Epoch: 0170 loss_train: 0.4350 acc_train: 0.9500 loss_val: 0.7218 acc_val: 0.7900 time: 0.0292s
Epoch: 0171 loss_train: 0.4388 acc_train: 0.9214 loss_val: 0.7220 acc_val: 0.7933 time: 0.0234s
Epoch: 0172 loss_train: 0.4233 acc_train: 0.9143 loss_val: 0.7209 acc_val: 0.7933 time: 0.0391s
Epoch: 0173 loss_train: 0.4127 acc_train: 0.9214 loss_val: 0.7194 acc_val: 0.7967 time: 0.0232s
Epoch: 0174 loss_train: 0.4109 acc_train: 0.9214 loss_val: 0.7175 acc_val: 0.8000 time: 0.0248s
Epoch: 0175 loss_train: 0.4007 acc_train: 0.9429 loss_val: 0.7153 acc_val: 0.8000 time: 0.0348s
Epoch: 0176 loss_train: 0.4488 acc_train: 0.9286 loss_val: 0.7146 acc_val: 0.8000 time: 0.0574s
Epoch: 0177 loss_train: 0.4228 acc_train: 0.9286 loss_val: 0.7127 acc_val: 0.8000 time: 0.0328s
Epoch: 0178 loss_train: 0.4102 acc_train: 0.9571 loss_val: 0.7113 acc_val: 0.8000 time: 0.0308s
Epoch: 0179 loss_train: 0.4740 acc_train: 0.9143 loss_val: 0.7098 acc_val: 0.8033 time: 0.0381s
Epoch: 0180 loss_train: 0.3849 acc_train: 0.9214 loss_val: 0.7100 acc_val: 0.8067 time: 0.0255s
Epoch: 0181 loss_train: 0.4159 acc_train: 0.9357 loss_val: 0.7088 acc_val: 0.8067 time: 0.0326s
Epoch: 0182 loss_train: 0.4165 acc_train: 0.9429 loss_val: 0.7073 acc_val: 0.8067 time: 0.0217s
Epoch: 0183 loss_train: 0.3979 acc_train: 0.9500 loss_val: 0.7060 acc_val: 0.8067 time: 0.0231s
Epoch: 0184 loss_train: 0.4081 acc_train: 0.9286 loss_val: 0.7041 acc_val: 0.8100 time: 0.0218s
Epoch: 0185 loss_train: 0.4149 acc_train: 0.9357 loss_val: 0.7030 acc_val: 0.8100 time: 0.0217s
Epoch: 0186 loss_train: 0.4190 acc_train: 0.9500 loss_val: 0.7022 acc_val: 0.8033 time: 0.0229s
Epoch: 0187 loss_train: 0.4182 acc_train: 0.9357 loss_val: 0.7005 acc_val: 0.8033 time: 0.0257s
Epoch: 0188 loss_train: 0.3929 acc_train: 0.9500 loss_val: 0.7007 acc_val: 0.8033 time: 0.0208s
Epoch: 0189 loss_train: 0.4057 acc_train: 0.9500 loss_val: 0.6997 acc_val: 0.8000 time: 0.0237s
Epoch: 0190 loss_train: 0.3713 acc_train: 0.9429 loss_val: 0.6994 acc_val: 0.8000 time: 0.0218s
Epoch: 0191 loss_train: 0.4061 acc_train: 0.9429 loss_val: 0.6990 acc_val: 0.8000 time: 0.0205s
Epoch: 0192 loss_train: 0.4258 acc_train: 0.9286 loss_val: 0.6983 acc_val: 0.8000 time: 0.0236s
Epoch: 0193 loss_train: 0.4071 acc_train: 0.9286 loss_val: 0.6980 acc_val: 0.8000 time: 0.0499s
Epoch: 0194 loss_train: 0.3884 acc_train: 0.9357 loss_val: 0.6974 acc_val: 0.8000 time: 0.0287s
Epoch: 0195 loss_train: 0.3809 acc_train: 0.9571 loss_val: 0.6968 acc_val: 0.8100 time: 0.0193s
Epoch: 0196 loss_train: 0.3758 acc_train: 0.9571 loss_val: 0.6969 acc_val: 0.8033 time: 0.0226s
Epoch: 0197 loss_train: 0.3804 acc_train: 0.9571 loss_val: 0.6961 acc_val: 0.8000 time: 0.0254s
Epoch: 0198 loss_train: 0.4148 acc_train: 0.9429 loss_val: 0.6962 acc_val: 0.8000 time: 0.0237s
Epoch: 0199 loss_train: 0.4131 acc_train: 0.9429 loss_val: 0.6963 acc_val: 0.8000 time: 0.0301s
Epoch: 0200 loss_train: 0.4209 acc_train: 0.9286 loss_val: 0.6971 acc_val: 0.8000 time: 0.0246s
Optimization Finished!
Total time elapsed: 4.7960s
Test set results: loss= 0.7245 accuracy= 0.8130
(base) rahuImaddula@rahuls-Air pygcn %
```

idx\_train=240

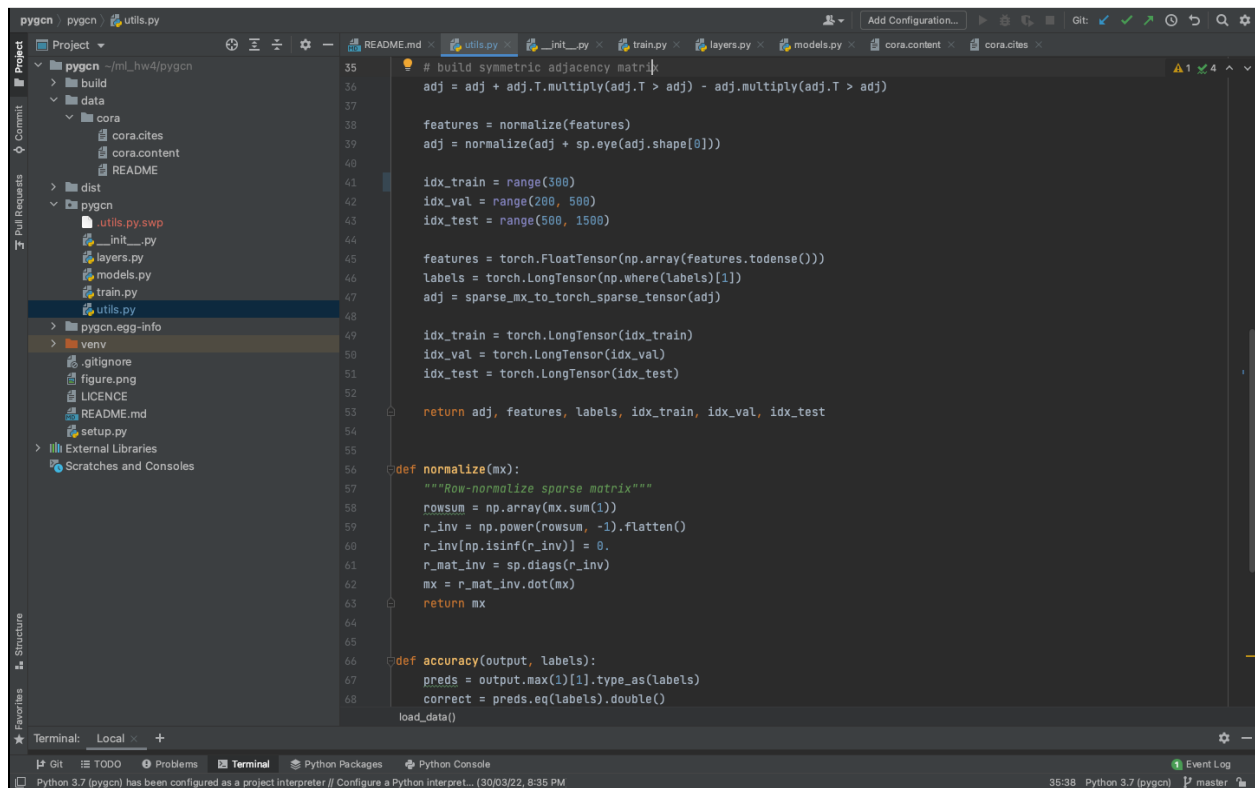
```
pygcn  pygcn  utils.py
Project
pygcn ~/ml_hw4/pygcn
> build
> data
  > cora
    cora.cites
    cora.content
    README
  > dist
  > pygcn
    utils.py.swp
    __init__.py
    layers.py
    models.py
    train.py
    utils.py
> pygcn.egg-info
> venv
  .gitignore
  figure.png
  LICENCE
  README.md
  setup.py
> External Libraries
Scratches and Consoles
Terminal: Local +
35 # build symmetric adjacency matrix
36 adj = adj + adj.T.multiply(adj.T > adj) - adj.multiply(adj.T > adj)
37
38 features = normalize(features)
39 adj = normalize(adj + sp.eye(adj.shape[0]))
40
41 idx_train = range(240)
42 idx_val = range(280, 500)
43 idx_test = range(500, 1500)
44
45 features = torch.FloatTensor(np.array(features.todense()))
46 labels = torch.LongTensor(np.where(labels)[1])
47 adj = sparse_mx_to_torch_sparse_tensor(adj)
48
49 idx_train = torch.LongTensor(idx_train)
50 idx_val = torch.LongTensor(idx_val)
51 idx_test = torch.LongTensor(idx_test)
52
53 return adj, features, labels, idx_train, idx_val, idx_test
54
55
56 def normalize(mx):
57     """Row-normalize sparse matrix"""
58     rowsum = np.array(mx.sum(1))
59     r_inv = np.power(rowsum, -1).flatten()
60     r_inv[np.isinf(r_inv)] = 0.
61     r_mat_inv = sp.diags(r_inv)
62     mx = r_mat_inv.dot(mx)
63     return mx
64
65
66 def accuracy(output, labels):
67     preds = output.max(1)[1].type_as(labels)
68     correct = preds.eq(labels).double()
69     load_data()
```

Output:



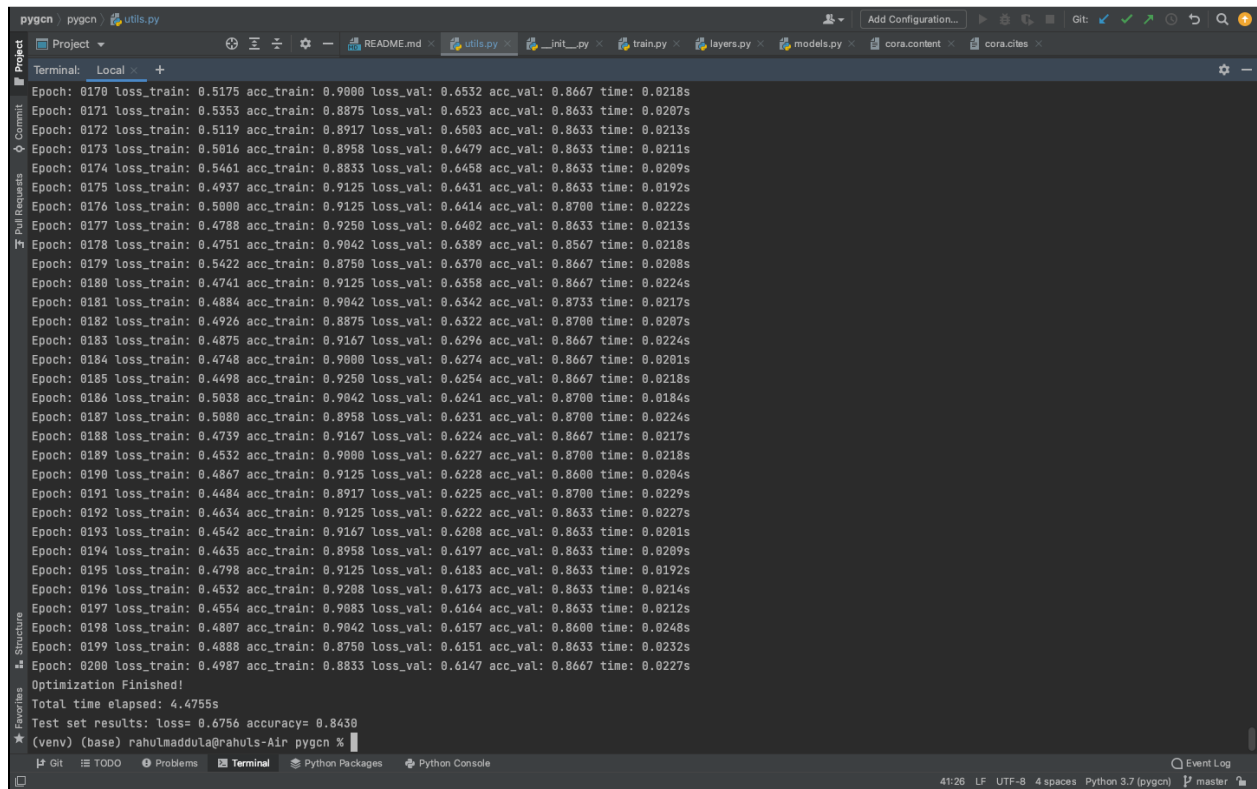
```
pygcn | pygcn | utils.py
Terminal: Local +
Epoch: 0170 loss_train: 0.5702 acc_train: 0.8786 loss_val: 0.7922 acc_val: 0.8233 time: 0.0242s
Epoch: 0171 loss_train: 0.5634 acc_train: 0.9080 loss_val: 0.7892 acc_val: 0.8267 time: 0.0212s
Epoch: 0172 loss_train: 0.5479 acc_train: 0.9286 loss_val: 0.7868 acc_val: 0.8200 time: 0.0235s
Epoch: 0173 loss_train: 0.5548 acc_train: 0.8929 loss_val: 0.7843 acc_val: 0.8200 time: 0.0228s
Epoch: 0174 loss_train: 0.5366 acc_train: 0.9357 loss_val: 0.7832 acc_val: 0.8067 time: 0.0215s
Epoch: 0175 loss_train: 0.5251 acc_train: 0.9143 loss_val: 0.7819 acc_val: 0.8033 time: 0.0238s
Epoch: 0176 loss_train: 0.5418 acc_train: 0.9071 loss_val: 0.7795 acc_val: 0.8033 time: 0.0217s
Epoch: 0177 loss_train: 0.5410 acc_train: 0.9080 loss_val: 0.7768 acc_val: 0.8033 time: 0.0235s
Epoch: 0178 loss_train: 0.5429 acc_train: 0.8929 loss_val: 0.7736 acc_val: 0.8033 time: 0.0252s
Epoch: 0179 loss_train: 0.5054 acc_train: 0.9080 loss_val: 0.7699 acc_val: 0.8033 time: 0.0228s
Epoch: 0180 loss_train: 0.4615 acc_train: 0.9214 loss_val: 0.7671 acc_val: 0.8100 time: 0.0239s
Epoch: 0181 loss_train: 0.5215 acc_train: 0.9071 loss_val: 0.7656 acc_val: 0.8067 time: 0.0229s
Epoch: 0182 loss_train: 0.5321 acc_train: 0.9080 loss_val: 0.7643 acc_val: 0.8033 time: 0.0212s
Epoch: 0183 loss_train: 0.4997 acc_train: 0.9214 loss_val: 0.7627 acc_val: 0.8067 time: 0.0207s
Epoch: 0184 loss_train: 0.5541 acc_train: 0.8643 loss_val: 0.7614 acc_val: 0.8100 time: 0.0222s
Epoch: 0185 loss_train: 0.5076 acc_train: 0.9143 loss_val: 0.7588 acc_val: 0.8067 time: 0.0228s
Epoch: 0186 loss_train: 0.5397 acc_train: 0.9143 loss_val: 0.7558 acc_val: 0.8167 time: 0.0225s
Epoch: 0187 loss_train: 0.5437 acc_train: 0.8929 loss_val: 0.7529 acc_val: 0.8167 time: 0.0224s
Epoch: 0188 loss_train: 0.5491 acc_train: 0.9071 loss_val: 0.7512 acc_val: 0.8233 time: 0.0224s
Epoch: 0189 loss_train: 0.5019 acc_train: 0.9080 loss_val: 0.7504 acc_val: 0.8100 time: 0.0228s
Epoch: 0190 loss_train: 0.4854 acc_train: 0.9071 loss_val: 0.7505 acc_val: 0.8067 time: 0.0229s
Epoch: 0191 loss_train: 0.5260 acc_train: 0.9143 loss_val: 0.7496 acc_val: 0.8033 time: 0.0238s
Epoch: 0192 loss_train: 0.5613 acc_train: 0.8357 loss_val: 0.7482 acc_val: 0.8033 time: 0.0226s
Epoch: 0193 loss_train: 0.4865 acc_train: 0.9357 loss_val: 0.7459 acc_val: 0.8033 time: 0.0229s
Epoch: 0194 loss_train: 0.5178 acc_train: 0.9071 loss_val: 0.7440 acc_val: 0.8033 time: 0.0222s
Epoch: 0195 loss_train: 0.5284 acc_train: 0.8929 loss_val: 0.7420 acc_val: 0.8033 time: 0.0222s
Epoch: 0196 loss_train: 0.4465 acc_train: 0.9286 loss_val: 0.7398 acc_val: 0.8033 time: 0.0223s
Epoch: 0197 loss_train: 0.4960 acc_train: 0.9143 loss_val: 0.7375 acc_val: 0.8067 time: 0.0226s
Epoch: 0198 loss_train: 0.4950 acc_train: 0.9071 loss_val: 0.7355 acc_val: 0.8167 time: 0.0225s
Epoch: 0199 loss_train: 0.4911 acc_train: 0.9080 loss_val: 0.7342 acc_val: 0.8200 time: 0.0219s
Epoch: 0200 loss_train: 0.4935 acc_train: 0.9214 loss_val: 0.7330 acc_val: 0.8267 time: 0.0225s
Optimization Finished!
Total time elapsed: 4.7067s
Test set results: loss= 0.7793 accuracy= 0.8250
(base) rahuImaddula@rahuls-Air pygcn %
```

idx\_train=300



```
pygcn | pygcn | utils.py
Project | pygcn | README.md | utils.py | __init__.py | train.py | layers.py | models.py | cora.content | cora.cites
Terminal: Local +
35 # build symmetric adjacency matrix
36 adj = adj + adj.T.multiply(adj.T > adj) - adj.multiply(adj.T > adj)
37
38 features = normalize(features)
39 adj = normalize(adj + sp.eye(adj.shape[0]))
40
41 idx_train = range(300)
42 idx_val = range(200, 500)
43 idx_test = range(500, 1500)
44
45 features = torch.FloatTensor(np.array(features.todense()))
46 labels = torch.LongTensor(np.where(labels)[1])
47 adj = sparse_mx_to_torch_sparse_tensor(adj)
48
49 idx_train = torch.LongTensor(idx_train)
50 idx_val = torch.LongTensor(idx_val)
51 idx_test = torch.LongTensor(idx_test)
52
53 return adj, features, labels, idx_train, idx_val, idx_test
54
55
56 def normalize(mx):
57     """Row-normalize sparse matrix"""
58     rowsum = np.array(mx.sum(1))
59     r_inv = np.power(rowsum, -1).flatten()
60     r_inv[np.isinf(r_inv)] = 0.
61     r_mat_inv = sp.diags(r_inv)
62     mx = r_mat_inv.dot(mx)
63     return mx
64
65
66 def accuracy(output, labels):
67     preds = output.max(1)[1].type_as(labels)
68     correct = preds.eq(labels).double()
69     load_data()
```

Output:



The screenshot shows a PyCharm IDE with a terminal window open. The terminal displays the output of a training process, showing metrics for each epoch from 170 to 200. The metrics include loss\_train, acc\_train, loss\_val, acc\_val, and time. The training process concludes with a message 'Optimization Finished!' and a summary of the total time elapsed and test set results.

```
pygcn | pygcn | utils.py
Project | README.md | utils.py | _init_.py | train.py | layers.py | models.py | cora.content | cora.cites
Terminal: Local +
Epoch: 0170 loss_train: 0.5175 acc_train: 0.9080 loss_val: 0.6532 acc_val: 0.8667 time: 0.0218s
Epoch: 0171 loss_train: 0.5353 acc_train: 0.8875 loss_val: 0.6523 acc_val: 0.8633 time: 0.0207s
Epoch: 0172 loss_train: 0.5119 acc_train: 0.8917 loss_val: 0.6503 acc_val: 0.8633 time: 0.0213s
Epoch: 0173 loss_train: 0.5016 acc_train: 0.8958 loss_val: 0.6479 acc_val: 0.8633 time: 0.0211s
Epoch: 0174 loss_train: 0.5461 acc_train: 0.8833 loss_val: 0.6458 acc_val: 0.8633 time: 0.0209s
Epoch: 0175 loss_train: 0.4937 acc_train: 0.9125 loss_val: 0.6431 acc_val: 0.8633 time: 0.0192s
Epoch: 0176 loss_train: 0.5000 acc_train: 0.9125 loss_val: 0.6414 acc_val: 0.8700 time: 0.0222s
Epoch: 0177 loss_train: 0.4788 acc_train: 0.9250 loss_val: 0.6402 acc_val: 0.8633 time: 0.0213s
Epoch: 0178 loss_train: 0.4751 acc_train: 0.9042 loss_val: 0.6389 acc_val: 0.8567 time: 0.0218s
Epoch: 0179 loss_train: 0.5422 acc_train: 0.8750 loss_val: 0.6370 acc_val: 0.8667 time: 0.0208s
Epoch: 0180 loss_train: 0.4741 acc_train: 0.9125 loss_val: 0.6358 acc_val: 0.8667 time: 0.0224s
Epoch: 0181 loss_train: 0.4884 acc_train: 0.9042 loss_val: 0.6342 acc_val: 0.8733 time: 0.0217s
Epoch: 0182 loss_train: 0.4926 acc_train: 0.8875 loss_val: 0.6322 acc_val: 0.8700 time: 0.0207s
Epoch: 0183 loss_train: 0.4875 acc_train: 0.9167 loss_val: 0.6296 acc_val: 0.8667 time: 0.0224s
Epoch: 0184 loss_train: 0.4748 acc_train: 0.9000 loss_val: 0.6274 acc_val: 0.8667 time: 0.0201s
Epoch: 0185 loss_train: 0.4498 acc_train: 0.9250 loss_val: 0.6254 acc_val: 0.8667 time: 0.0218s
Epoch: 0186 loss_train: 0.5038 acc_train: 0.9042 loss_val: 0.6241 acc_val: 0.8700 time: 0.0184s
Epoch: 0187 loss_train: 0.5080 acc_train: 0.8958 loss_val: 0.6231 acc_val: 0.8700 time: 0.0224s
Epoch: 0188 loss_train: 0.4739 acc_train: 0.9167 loss_val: 0.6224 acc_val: 0.8667 time: 0.0217s
Epoch: 0189 loss_train: 0.4532 acc_train: 0.9000 loss_val: 0.6227 acc_val: 0.8700 time: 0.0218s
Epoch: 0190 loss_train: 0.4867 acc_train: 0.9125 loss_val: 0.6228 acc_val: 0.8600 time: 0.0204s
Epoch: 0191 loss_train: 0.4484 acc_train: 0.8917 loss_val: 0.6225 acc_val: 0.8700 time: 0.0229s
Epoch: 0192 loss_train: 0.4634 acc_train: 0.9125 loss_val: 0.6222 acc_val: 0.8633 time: 0.0227s
Epoch: 0193 loss_train: 0.4542 acc_train: 0.9167 loss_val: 0.6208 acc_val: 0.8633 time: 0.0201s
Epoch: 0194 loss_train: 0.4635 acc_train: 0.8958 loss_val: 0.6197 acc_val: 0.8633 time: 0.0209s
Epoch: 0195 loss_train: 0.4798 acc_train: 0.9125 loss_val: 0.6183 acc_val: 0.8633 time: 0.0192s
Epoch: 0196 loss_train: 0.4532 acc_train: 0.9208 loss_val: 0.6173 acc_val: 0.8633 time: 0.0214s
Epoch: 0197 loss_train: 0.4554 acc_train: 0.9083 loss_val: 0.6164 acc_val: 0.8633 time: 0.0212s
Epoch: 0198 loss_train: 0.4807 acc_train: 0.9042 loss_val: 0.6157 acc_val: 0.8600 time: 0.0248s
Epoch: 0199 loss_train: 0.4888 acc_train: 0.8750 loss_val: 0.6151 acc_val: 0.8633 time: 0.0232s
Epoch: 0200 loss_train: 0.4987 acc_train: 0.8833 loss_val: 0.6147 acc_val: 0.8667 time: 0.0227s
Optimization Finished!
Total time elapsed: 4.4755s
Test set results: loss=0.6756 accuracy=0.8430
* (venv) (base) rahulmaddula@rahuls-Air pygcn %
```

41:26 LF UTF-8 4 spaces Python 3.7 (pygcn) master