Programming Assignment

Trina Dutta
Spencer Cain

CIS 5930: Graph Neural Networks

July 19, 2022

Graph Generation

In order to generate the graph, we created an edge list such that an edge exists if a node is among the seven nearest neighbors of another node. To find the seven nearest neighbors, we used the k-nearest-neighbor algorithm with the distance measure based on the correlation of each image. This produces an undirected graph with 9298 nodes and 64001 edges. The supernode adjacency matrix is defined as

$$\mathbf{A}_{SN} = \begin{bmatrix} 2612 & 335 & 2380 & 910 & 2678 & 327 & 141 & 429 & 386 & 329 \\ 8747 & 0 & 3 & 1 & 7 & 1 & 0 & 0 & 89 & 0 \\ 3857 & 0 & 111 & 545 & 790 & 323 & 14 & 81 & 379 & 220 \\ 4101 & 0 & 303 & 9 & 856 & 239 & 19 & 6 & 95 & 29 \\ 4891 & 0 & 320 & 248 & 5 & 272 & 58 & 5 & 45 & 3 \\ 2729 & 0 & 523 & 224 & 996 & 160 & 30 & 30 & 147 & 70 \\ 3171 & 3 & 93 & 507 & 461 & 110 & 12 & 90 & 927 & 424 \\ 4807 & 0 & 159 & 20 & 35 & 209 & 56 & 0 & 186 & 0 \\ 4034 & 0 & 86 & 80 & 465 & 82 & 82 & 32 & 25 & 36 \\ 5012 & 0 & 172 & 15 & 114 & 211 & 147 & 0 & 30 & 0 \end{bmatrix}$$

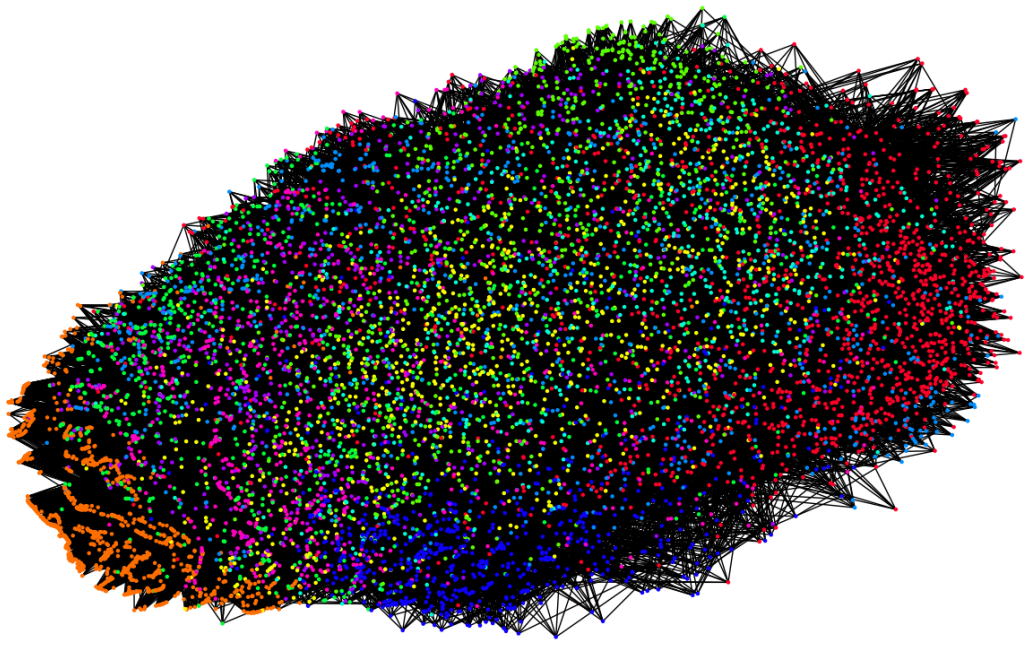

Figure 1: The graph generated using the KNN algorithm based on the correlation between images.

Task I – Graph Neural Network Design

We made the choice to create similar model architectures in order to evaluate the dynamics of each model comparatively. Since GNN models typically perform best with two filtering layers, and convolutional blocks are similar to the GNN filtering layer, we chose for each model to have two filter-type layers. The aggregation layer of the CNN is not included in the block, so we add max-pooling. Finally, we pass the flattened outputs of the filtering layers into a dense layer with an output dimension of 64 and a dense layer with an output dimension of 10, the number of classes. Each block/layer also has a non-linear activation for SGD purposes.

CNN Model

The CNN model consists of two convolutional layers and two fully connected layers. The organization of the model is as follows:

Listing 1: CNN Model

```
cnn.Classifier(
  Conv2d(1, 32, kernel\_size=3)
  MaxPool2d(2, 2)
  Dropout2d(0.3)
  Conv2d(32, 64, kernel\_size=3)
  MaxPool2d(2, 2)
  Dropout2d(0.3)
  Linear(256, 64)
  ReLU()
  Linear(64, 10)
)
```

Spectral-Based GNN Model

The spectral-based GNN model consists of two Cheby filter layers and two fully connected layers. The organization of the model is as follows:

Listing 2: Spectral-Based GNN Model

```
spectral.Classifier(
  (gc1): DenseChebConv()
  (elu): ELU(alpha=1.0)
  (dropout): Dropout(p=0.5, inplace=False)
  (gc2): DenseChebConv()
  (elu): ELU(alpha=1.0)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=256, out_features=64, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```

Spatial-Based GNN Model

The spatial-based GNN model consists of two graph attention layers and two fully connected layers. The organization of the model is as follows:

Listing 3: Spatial-Based GNN Model

```
spatial.Classifier(
  (gat1): GATConv(
    (fc): Linear(in_features=256, out_features=768, bias=False)
    (feat_drop): Dropout(p=0.0, inplace=False)
    (attn_drop): Dropout(p=0.5, inplace=False)
    (leaky_relu): LeakyReLU(negative_slope=0.2)
    (res_fc): Linear(in_features=256, out_features=768, bias=False)
  )
```

```
  (elu): ELU(alpha=1.0)
  (dropout): Dropout(p=0.5, inplace=False)
  (gat2): GATConv(
    (fc): Linear(in_features=256, out_features=768, bias=False)
    (feat_drop): Dropout(p=0.0, inplace=False)
    (attn_drop): Dropout(p=0.5, inplace=False)
    (leaky_relu): LeakyReLU(negative_slope=0.2)
    (res_fc): Linear(in_features=256, out_features=768, bias=False)
  )
  (elu): ELU(alpha=1.0)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=2304, out_features=64, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```

Task II - Node Classification Techniques and Analyses

DL Method

After completing a hyper-parameter search, we found that the CNN model performs best on the test set with a batch size of 32 and a learning rate of 0.01 over 6 epochs. The model is trained using the Adam optimizer and cross entropy loss. With best hyperparemeters, the model achieves accuracy of 1.0 on the test set. Below are the loss and accuracy curves on the test set during training.

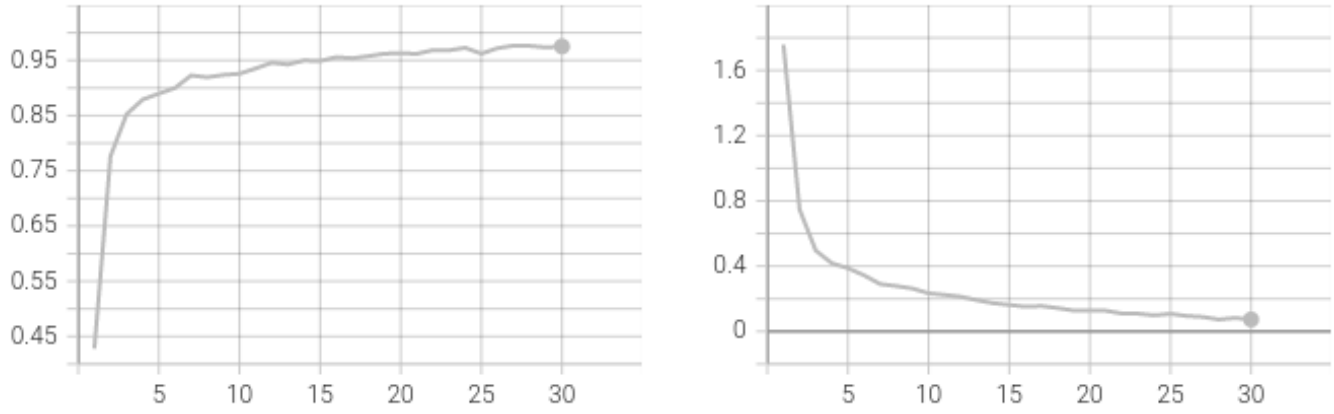

Figure 2: Train accuracy and cross entropy loss.

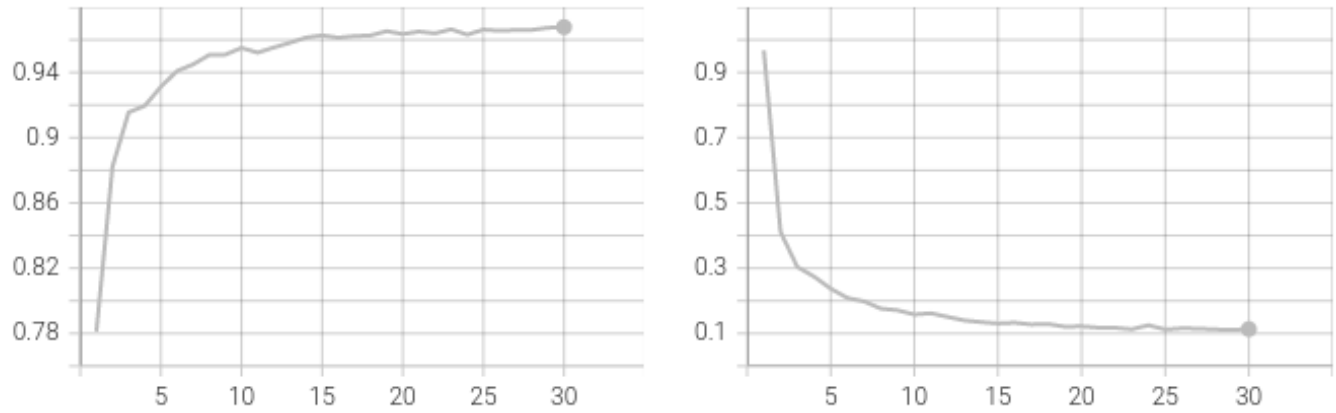

Figure 3: Test accuracy and cross entropy loss.

Here is the confusion matrix on the test set using the trained model:

$$\begin{bmatrix} 1179 & 1 & 5 & 1 & 0 & 3 & 2 & 0 & 3 & 0 \\ 0 & 1004 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 3 & 691 & 2 & 7 & 3 & 2 & 6 & 15 & 0 \\ 0 & 1 & 2 & 638 & 0 & 10 & 0 & 1 & 5 & 1 \\ 3 & 4 & 0 & 0 & 616 & 1 & 8 & 1 & 11 & 8 \\ 1 & 0 & 4 & 8 & 1 & 529 & 7 & 0 & 4 & 2 \\ 6 & 5 & 2 & 0 & 3 & 2 & 645 & 0 & 1 & 0 \\ 6 & 0 & 7 & 0 & 1 & 0 & 0 & 619 & 2 & 10 \\ 1 & 4 & 0 & 3 & 3 & 4 & 1 & 2 & 521 & 3 \\ 2 & 0 & 1 & 0 & 5 & 5 & 0 & 9 & 1 & 621 \end{bmatrix}$$

C&S Method

Using the C&S method conveniently provided in the Torch Geometric package, we correct-and-smoothed the output of the CNN to a test accuracy of 0.99822. In order to do so, we used 50 node correcting iterations and 50 node smoothing iterations, with their related αs set to 1. Here is the confusion matrix on the test set using the trained model:

$$\begin{bmatrix} 1193 & 0 & 1 & 3 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 1005 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 730 & 1 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 654 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 649 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 556 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 663 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 643 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 542 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 643 \end{bmatrix}$$

Spectral GNN Method

After completing a hyper-parameter search, we found that the spectral-based GNN model performs best on the test set with a learning rate of 0.005 over 17 epochs. The model is trained using the Adam optimizer and cross entropy loss. With best hyperparemeters, the model achieves accuracy of 92.03% on the test set. Below are the loss and accuracy curves on the test set during training.

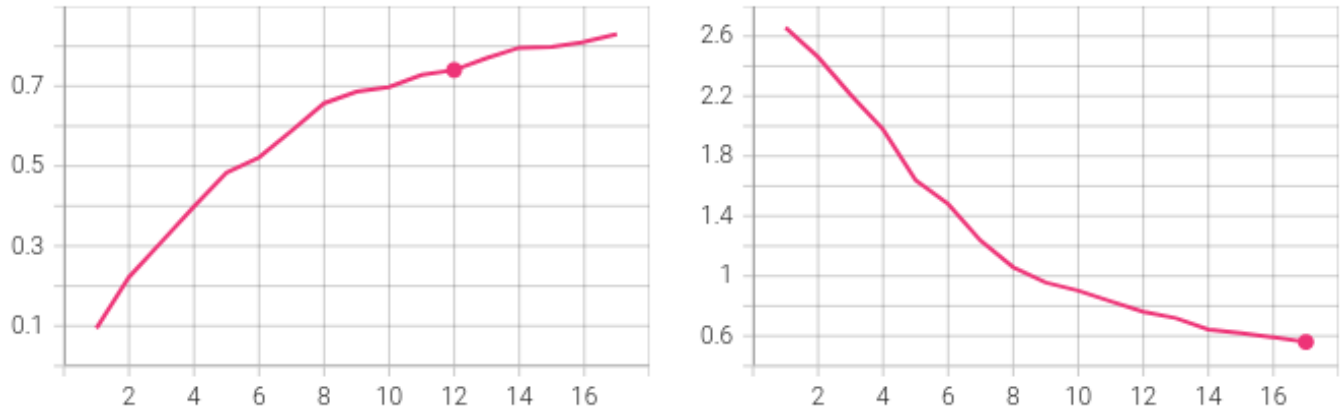

Figure 4: Train accuracy and cross entropy loss.

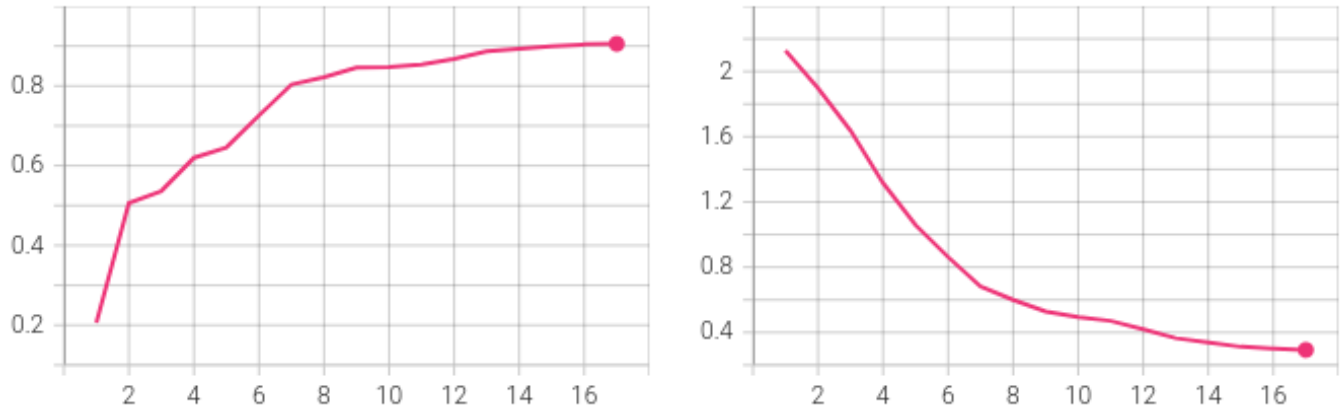

Figure 5: Test accuracy and cross entropy loss.

Here is the confusion matrix on the test set using the trained model:

$$\begin{bmatrix} 1154 & 0 & 3 & 2 & 7 & 16 & 4 & 0 & 8 & 0 \\ 0 & 1003 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 11 & 1 & 663 & 13 & 21 & 3 & 2 & 7 & 10 & 0 \\ 1 & 1 & 8 & 609 & 1 & 20 & 0 & 7 & 10 & 1 \\ 0 & 14 & 14 & 0 & 558 & 0 & 6 & 3 & 6 & 21 \\ 21 & 2 & 9 & 26 & 17 & 461 & 11 & 1 & 5 & 3 \\ 15 & 8 & 26 & 0 & 5 & 5 & 602 & 0 & 3 & 0 \\ 0 & 1 & 4 & 0 & 1 & 3 & 0 & 617 & 4 & 15 \\ 5 & 10 & 9 & 25 & 13 & 8 & 1 & 4 & 458 & 9 \\ 0 & 5 & 3 & 2 & 62 & 2 & 0 & 47 & 5 & 518 \end{bmatrix}$$

Spatial GNN Method

After completing a hyper-parameter search, we found that the spatial-based GNN model performs best on the test set with a learning rate of 0.005 over 30 epochs. The model is trained using the Adam optimizer and cross entropy loss. With best hyperparemeters, the model achieves accuracy of 92.54% on the test set. Below are the loss and accuracy curves on the test set during training.

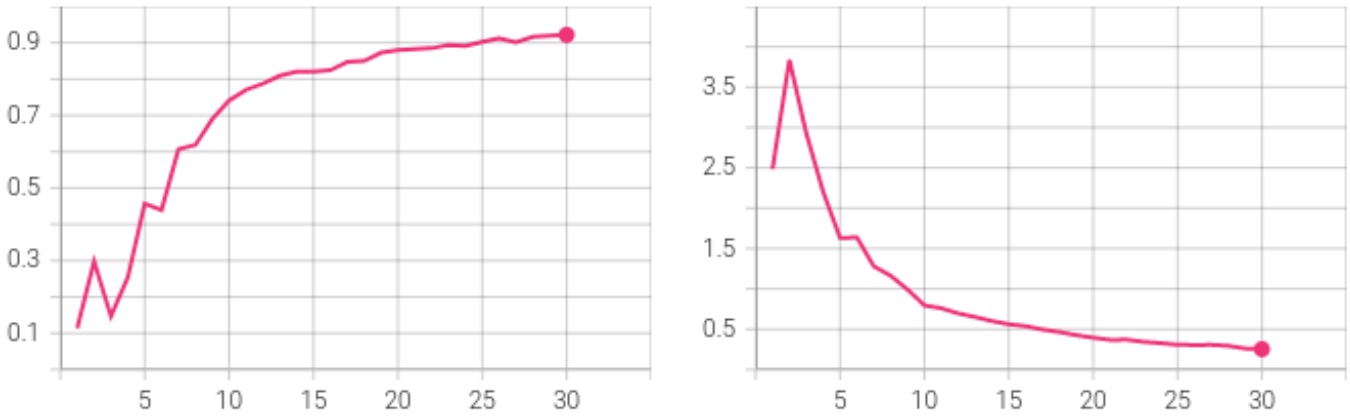

Figure 6: Train accuracy and cross entropy loss.

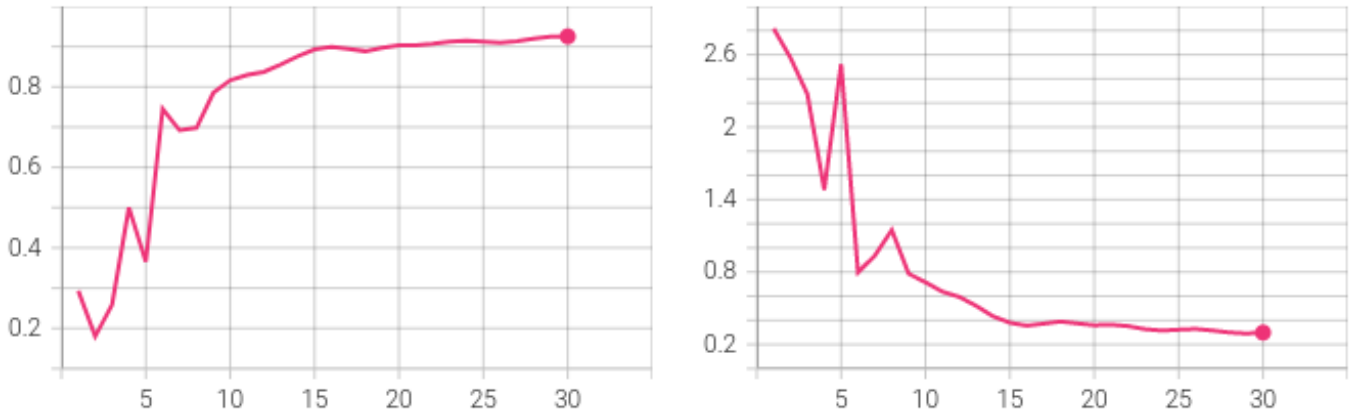

Figure 7: Test accuracy and cross entropy loss.

Here is the confusion matrix on the test set using the trained model:

$$\begin{bmatrix} 1138 & 0 & 23 & 0 & 4 & 20 & 3 & 0 & 6 & 0 \\ 0 & 997 & 0 & 1 & 0 & 1 & 0 & 1 & 5 & 0 \\ 4 & 2 & 675 & 10 & 13 & 6 & 3 & 3 & 15 & 0 \\ 2 & 0 & 20 & 577 & 1 & 22 & 0 & 3 & 31 & 2 \\ 1 & 20 & 6 & 0 & 583 & 2 & 5 & 0 & 12 & 23 \\ 12 & 0 & 5 & 24 & 15 & 485 & 4 & 0 & 9 & 2 \\ 9 & 7 & 8 & 0 & 3 & 7 & 623 & 0 & 7 & 0 \\ 2 & 3 & 0 & 1 & 4 & 1 & 0 & 614 & 5 & 15 \\ 2 & 1 & 9 & 3 & 5 & 6 & 3 & 2 & 511 & 0 \\ 0 & 2 & 4 & 1 & 10 & 1 & 0 & 20 & 16 & 590 \end{bmatrix}$$