A blue sky with white clouds

Description automatically generated



Artificial Neural Networks

||24CSCI33H

CW2-GNNs

|  |  |
| --- | --- |
| Name | ID |
| Omar Ayman | 227931 |
| Zeinab Donia | 222188 |
| Retaj Mohamed | 226532 |
| Konouz Abdelaziz | 227447 |

# Dataset

The TUDataset, imported from the Pytorch Geometric module, is a dataset consisting of 1113 graphs representing proteins where the nodes represent amino acids and the edges connect the nodes that have a distance less than 0.6nm. There are three features and two classes in the dataset. The task is to classify the proteins as enzymes or non-enzymes.

Data Preprocessing

## **Cayley Graph Propagation**

This approach is used to address the problem of over-squashing, which hinders message passing in nodes and causes loss of information. It creates virtual nodes to expand the graph while keeping the index of each edge in the original graph to reconstruct the graph after expansion where the virtual nodes are masked (zeroed out) . This doesn’t change the graph structure but only enhances the graph expressiveness in the spatial features and improves message propagation/passing. The “ExpanderTransform” class augments the graph data by embedding nodes and edges from a Cayley graph into the original graph. The Cayley graph contains at least as many nodes as the input graph and is saved in the memory to avoid recomputing for the same n.

A graph of a graph

Description automatically generated with medium confidence

Figure Before and After Caylee Graph Propagation

The Graph is larger as the distance between nodes is expanded, solving the problem of feature similarity where the nodes overlap and cannot be differentiated. After Cayley Propagation, the testing accuracy of the GCN model was 0.6858.

Some of the advantages of Cayley Graph Propagation include[1]:

1. Enhanced Graph Connectivity, where after augmenting the graph with additional nodes, this increases connectivity and enhances information flow. By providing multiple pathways to a node, this allows the nodes to become more distinct and mitigates over-smoothing.
2. Capture of complex relationships not evident in original graph as Cayley graphs help model regular and repeating patterns.
3. Robustness and Generalization as a more connected graph has improved performance in unseen data and is less sensitive to noise.

## **Data Augmentation**

We carried out feature augmentation (dropout) where we randomly zero out node features to add noise and simulate missing features, this helps the model to generalize better. After feature augmentation, the testing accuracy in GCN was 0.6947, which increased slightly than before. Previous augmentation methods, which added noise randomly, they also increased accuracy greatly but the results were not valid because adding random nodes did increase connectivity but changed the fundamental structure of the protein.

# Two Graph Neural Network Architectures

## **GCN**

The architecture has three Graph Convolutional Layers each with 64 hidden channels. Each GCNConv layer transforms and propagates node features by aggregating information from neighboring nodes. Batch normalization is applied after each convolutional layer to stabilize training and improve generalization. Then there’s a Readout layer that uses global mean pooling to aggregate node features into graph-level features by computing the mean of all node features in a graph. Furthermore, dropout is used as a regularization technique to prevent overfitting by randomly zeroing out nodes during training with a probability defined by dropout. Finally, there is a linear classification layer that maps the pooled graph embeddings to the number of classes.

## **GINs**

There are three GINConv layers, each aggregates information from a node's neighbors and updates the node's features. Each is made up of a linear layer to learn transformations, batch normalization to stabilize learning by normalizing feature distributions, a ReLU activation function.

# Different Hyperparameters

The base model parameters in both models are:

* Hidden Channels: 32
* Dropout: 0.4
* Learning Rate: 0.001
* Epochs: 171
* Batch Size: 32
* Weight Decay: 0.0005
* Optimizer: Adam
* Loss Function: CrossEntropyLoss

**GIN: Test Loss: 0.74 | Test Acc: 73.44%**

**GCN: Test Accuracy: 0.6858**

## **GCN**

The base model parameters are:

* Hidden Channels: 64
* Dropout: 0.5
* Learning Rate: 0.01
* Epochs: 171
* Batch Size: 64
* Weight Decay: 0.0001
* Optimizer: Adam
* Loss Function: CrossEntropyLoss

We carried out hyperparameter tuning using grid search where each combination of hyperparameters are evaluated and the best combination is selected. The selected hyperparameters were:

* Hidden Channels: 32
* Dropout: 0.3
* Learning Rate: 0.0001
* Epochs: 100
* Batch Size: 16
* Weight Decay: 0.0001
* Optimizer: Adam
* Loss Function: CrossEntropyLoss

## **GIN**

The base model parameters are:

* Dropout rate: 0.5
* Activation function: LeakyRelU
* Weight decay: 0.01
* Learning rate: 0.01
* Optimizer: Adam
* Number of Layers: 3
* Epochs: 100
* Loss Function: CrossEntropyLoss

After iterative testing of different variations of the hyperparameters, this gave the best accuracy of 76%:

* Dropout rate: 0.4
* Activation function: LeakyRelU
* Weight decay: 0.01
* Learning rate: 0.0001
* Optimizer: AdamW
* Number of Layers: 3
* Epochs: 100
* Loss Function: CrossEntropyLoss

Comparisons using different evaluation metrics- advantages and disadvantages

## **Learning Curve**

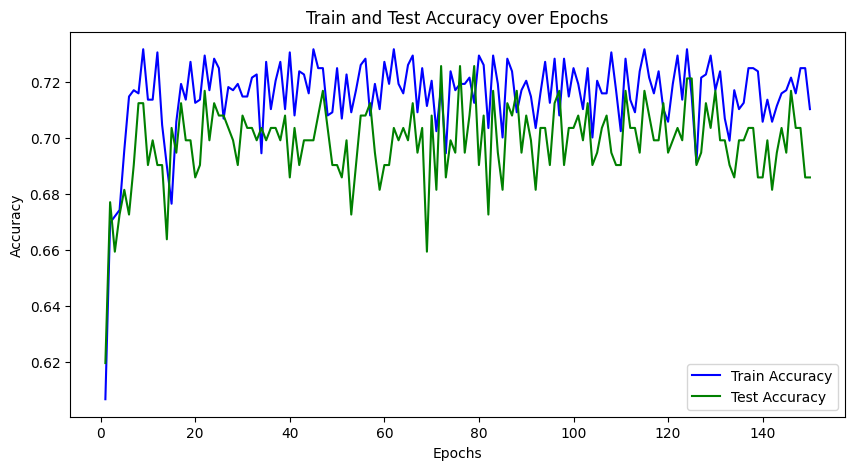


Figure GCN base model

Test Accuracy: 0.6858

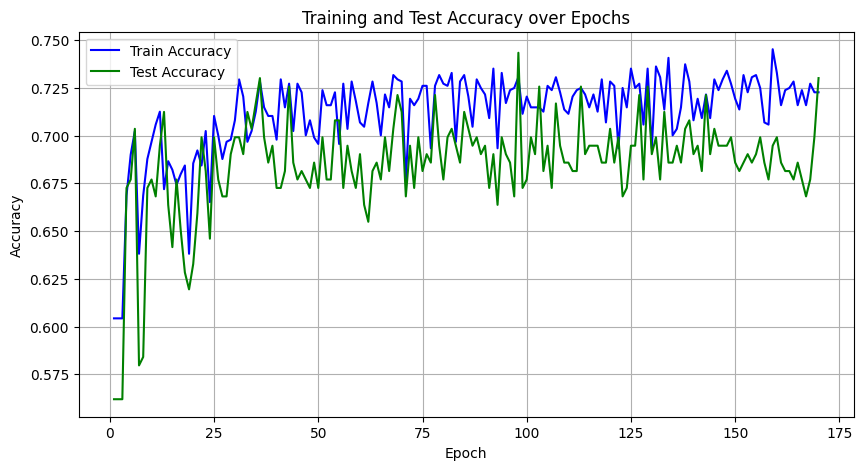


Figure GCN model one

0.7301

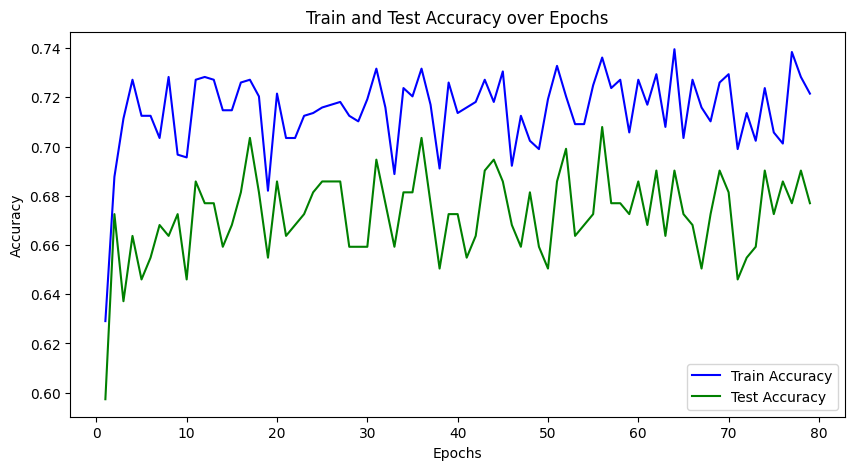


Figure GCN with hyperparameter tuning

0.6770

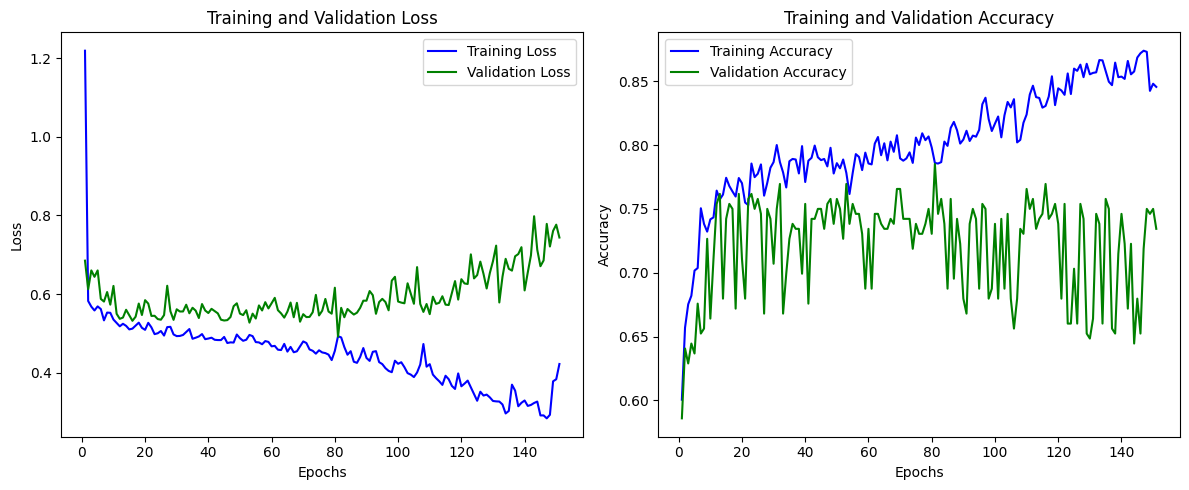


Figure GIN base model

Test Loss: 0.74 | Test Acc: 73.44%

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure GIN base model

Test Loss: 0.60 | Test Acc: 65.83%

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure GIN after adjusting hyperparameters

Test Loss: 0.62 | Test Acc: 71.67%

## **Accuracy**

# Conclusion

# References

[1] J. J. Wilson, M. Bechler-Speicher, and P. Veličković, “Cayley Graph Propagation,” Oct. 04, 2024, *arXiv*: arXiv:2410.03424. doi: 10.48550/arXiv.2410.03424.