Graph Neural Networks and its applications in IPR calculation in Complex Networks

Harish-Chandra Research Institute

Diffused, delocalized ,weakly localized and strongly localized are the different formats of information propagation in complex systems.The following study demonstrates a **graph convolution** , **graph convolution attention** and the **auto regressive moving average** neural network framework has been developed to identify the behaviour of linear dynamical system.The study portrays the fact that the trained model distinguishes the different states with high accuracy.In addition to the above , an analytical derivation has been provided for the forward and backward propagation and the superiority of the new model architecture of GCN contrary to the existing framework for ipr(inverse participation ratio analysis).

Supervisor: Chandrakala Meena (Professor, IISER Pune)
Author: Shikharkya Deb (MSc, HRI)

I INTRODUCTION

Diffused or delocalized, weakly localized and strongly localized are the three distinct states from information propagation in complex networks.Information condensed over a single component refer to strong localization while for few components refer to weak localization and for information diffusing evenly refer to the delocalization.In solid state physics localization and its presence or absence influences the properties of molecules or materials.
Network Centrality measure, spectral partitioning and development of approximation algorithms are the fundamental network problems.Investigation of the localization and delocalization behavior of complex networks is important for gaining insight into such fundamental network problems.In some cases non-linear systems can be solved by their transformation into linear counterparts near fixed points. Hence , understanding linear systems and their solutions is an essential first step toward comprehending more complex nonlinear dynamical systems.

In this study we develop Graph Neural Network (GNN) Architecture to identify the behavior of linear dynamical states on complex networks.Datasets for individual networks across topologies which include star,circle,wheel and scalefree and based on their feature weights training labels are derived from the inverse participation ratio (IPR) value of the principal eigen vector (PEV) of the network adjacency matrices. The GNN model takes in the network structure as input and predicts IPR values through a regression operation based on multi-layer perceptron (MLP), enabling the identification of graphs into their respective linear dynamical states.Our all the three models overall and also among themselves outperform each other in identifying different states and is particularly effective across varying sized networks.The key advantage of using GNN is its transferability, referring to its ability in getting trained on smaller networks and generalizing well to larger networks while testing.An analytical framework to understand the explainability of our model has been provided for establishment of the superiority of the models over each other.The models are also pretty well compatible and applicable to real world data sets.

II Problem Setup

Consider a linear dynamical process M taking place on unknown network structures represented as $G = (V, E)$ where $V = (v_1, v_2, v_3, ..., v_n)$ is the set of vertices(nodes),$E = (v_i, v_j)|v_i, v_j \epsilon V$ is the set of edges(connections).The degree of a node adjacent to it, which is given by $\Sigma_{j=1}^{n} a_{ij}$ where a_{ij} is the adjacency matrix element.The links in G represent dynamic interactions whose nature depends on context.For instance,a_{ij} captures a interaction between individuals i and j. The linear dynamics of node i is represented by :-

$$\frac{dx_i(t)}{dt} = \gamma x_i(t) + \zeta \sum_{j=1}^{n} a_{ij} x_j(t) \qquad (1)$$

where $x_i(t)$ is the dynamical term, the second term captures the neighboring interactions at time t and γ, ζ are the model parameters of the linear dynamical system.In matrix notation equation (1) can be expressed as :-

$$\frac{dx(t)}{dt} = Bx(t) \qquad (2)$$

Where $x(t) = (x_1(t), x_2(t), x_3(t).....x_n(t))^T$, $B = \gamma I + \zeta A$ is the transition, $\mathbf{A}$ is the adjacency and $\mathbf{I}$ is the identity matrices, respectively, the steady state behavior of the (x^*) of the linear dynamical system can be found as :-

$$x(t) = e^{Bt} x(0) \rightarrow x^* \sim u_1^B \qquad (3)$$

where u_1^B is the PEV of B. We consider $B = \mathbf{R}^{n \times n}$ is diagonalizable , $B = U \Lambda U^{-1}$ and $UU^{-1} = I$ where columns of U are the eigenvectors $(u_1^B, u_2^B, u_3^B, ..., u_n^B)$ of B and having n distinct eigenvalues which can be defined as $(\lambda_1^B, \lambda_2^B, \lambda_3^B, ..., \lambda_n^B)$.$x(0)$ being an initial state

, it can be defined as a combination of the eigenvalues and eigenvectors of $\mathbf{B}$, and therefore it can be written as :-

$$x(t) = e^{Bt}[c_1(0)u_1^B + c_2(0)u_2^B + c_3(0)u_3^B + ... + c_n(0)u_n^B]$$

$$x(t) = Ue^{\Lambda t}U^{-1}[c_1(0)u_1^B + c_2(0)u_2^B + ... + c_n(0)u_n^B] \quad (4)$$

$$x(t) = Ue^{\Lambda t}U^{-1}Uc(0) \quad (5)$$

$$x(t) = Ue^{\Lambda t}c(0) \quad (6)$$

$$x(t) = c_1(0)e^{\lambda_1^B t} + c_2(0)e^{\lambda_2^B t} + ... + c_n(0)e^{\lambda_n^B t} \quad (7)$$

$$x(t) = \Sigma_{i=1}^n c_i(0)e^{\lambda_i^M t u_i^M} \quad (8)$$

Here $c(0) = (c_1(0), c_2(0), c_3(0), ..., c_n(0))^T$ and the expression e^{Bt} is :-

$$e^{Bt} = I + Bt + \frac{(Bt)^2}{2} + \frac{(Bt)^3}{3} + ... \quad (9)$$

$$e^{Bt} = I + U\Lambda U^{-1}t + \frac{U\Lambda U^{-1}tU\Lambda U^{-1}t}{2!} + \frac{U\Lambda U^{-1}tU\Lambda U^{-1}tU\Lambda U^{-1}t}{3!} + ... \quad (10)$$

$$e^{Bt} = U[I + \Lambda t + \frac{(\Lambda t)^2}{2!} + \frac{(\Lambda t)^3}{3!} + ...]U^{-1} \quad (11)$$

$$e^{Bt} = Ue^{\Lambda t}U^{-1} \quad (12)$$

For $t \to \infty$, it can be approximated that equation (8) as :-

$$x^* \sim c_1(0)e^{\Lambda_1^{Bt}} \sim u_1^B \quad (13)$$

Since the largest eigenvalue dominates , λ_1 the principal eigenvector will decide the steady state behavior of the system.The localization and delocalization is quantified using the inverse participation ratio , which is the sum of the fourth power of state vector entries and is calculated using :-

$$y_{x*} = \frac{\Sigma_{i=1}^n x_i^{*4}}{[\Sigma_{i=1}^n x^{*2}]^2} \quad (14)$$

where x_i^* is the ith component of $x^* = (x_1^*, x_2^*, x_3^*...., x_n^*)^T$ and $\Sigma_{i=1}^n x_i^{*2}$ is the normalization term.

To understand the category of dynamical behavior of the states $x*$ a formulation can be devised for a threshold scheme for IPR identification within the boundary [1/n,1).
The defination includes two threshold values which are η_1 and η_2 with an additional parameter δ for flexibility in the boundaries.

Delocalized Region (r_1):-

$$r_1 = (y \quad \epsilon[1/n, 1) \leq \eta_1 - \delta) \quad (15)$$

Weakly localized region (r_2):-

$$r_2 = (y \quad \epsilon[1/n, 1) | \eta_1 - \delta < y < \eta_2 + \delta) \quad (16)$$

Strongly Localized Region (r_3):-

$$r_3 = (y \quad \epsilon[1/n, 1) | y \geq \eta_2 + \epsilon]) \quad (17)$$

The entire problem revolves around finding out a target value the IPR, the entire task is of graph regression for a set of graphs $(G_i)_{i=1}^N$ where each $G_i = (V_i, E_i)$ consists a set of nodes V_i and a set of edges E_i, where each graph is represented by its adjacency matrix A_i.For each graph G_i there is a target IPR value, $y^{(i)} \epsilon \mathbf{R}$ and for each node $v \epsilon V$ in G_i, there is an associated feature vector $h_j^{(i)} \epsilon \mathbf{R}^{|\mathbf{V_i}| \times \mathbf{d}}$ be the node feature matrix where $h_j^{(i)}$ is the jth row.The object is to learn a function $f : G \to \mathbf{R}$ such that $f(G_i) \approx y^{(i)}$.

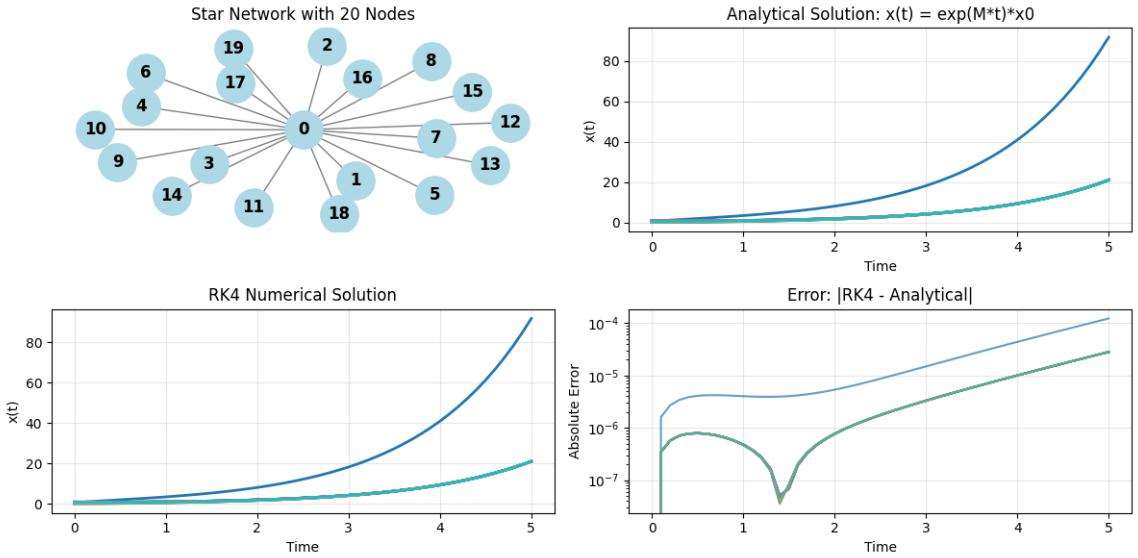

Figure 1: The time series analysis of dynamical equation for star network using RK-4 method

III Architectures and Methodologies

The function to be studied is parameterized using a Graph Convolutional Network in our model. The prediction for IPR in each of our graphs from G_i is denoted by $y_i = f(G_i, \alpha, \beta)$, where the model parameters α and β are learned by minimizing a loss function between the predicted values $\hat{y}^{(i)}$ and the true target values $y^{(i)}$. For classification tasks, the Cross Entropy loss is used, while for IPR value regression, the Mean Squared Error loss function is employed.

Cross Entropy Loss: This loss function is particularly useful for classification problems with C classes. If provided, the optional argument **weight** should be a 1D Tensor assigning weight to each class, which is especially beneficial for handling unbalanced training sets.

The *input* is expected to contain unnormalized logits for each class (which do not need to be positive or sum to 1). The input can be a Tensor of size C for unbatched input, or $(minibatch, C)$ or $(minibatch, C, d_1, d_2, \ldots, d_K)$ with $K \geq 1$ for higher-dimensional inputs, such as per-pixel cross entropy loss computation in 2D images.

The target should contain class indices in the range $[0, C)$. If *ignore_index* is specified, this loss also accepts this class index. The unreduced loss (with reduction set to 'none') can be described as: For ($y_n \neq$ ignore_index)

$$l(x, y) = L = (l_1, \ldots, l_N)^T, \quad l_n = -w_n \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot \mathbf{I} \tag{18}$$

where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \ldots, d_k$ for the K-dimensional case.

The MSE (Mean Squared Error) loss function is defined as:

$$L(\alpha) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})^2 \tag{19}$$

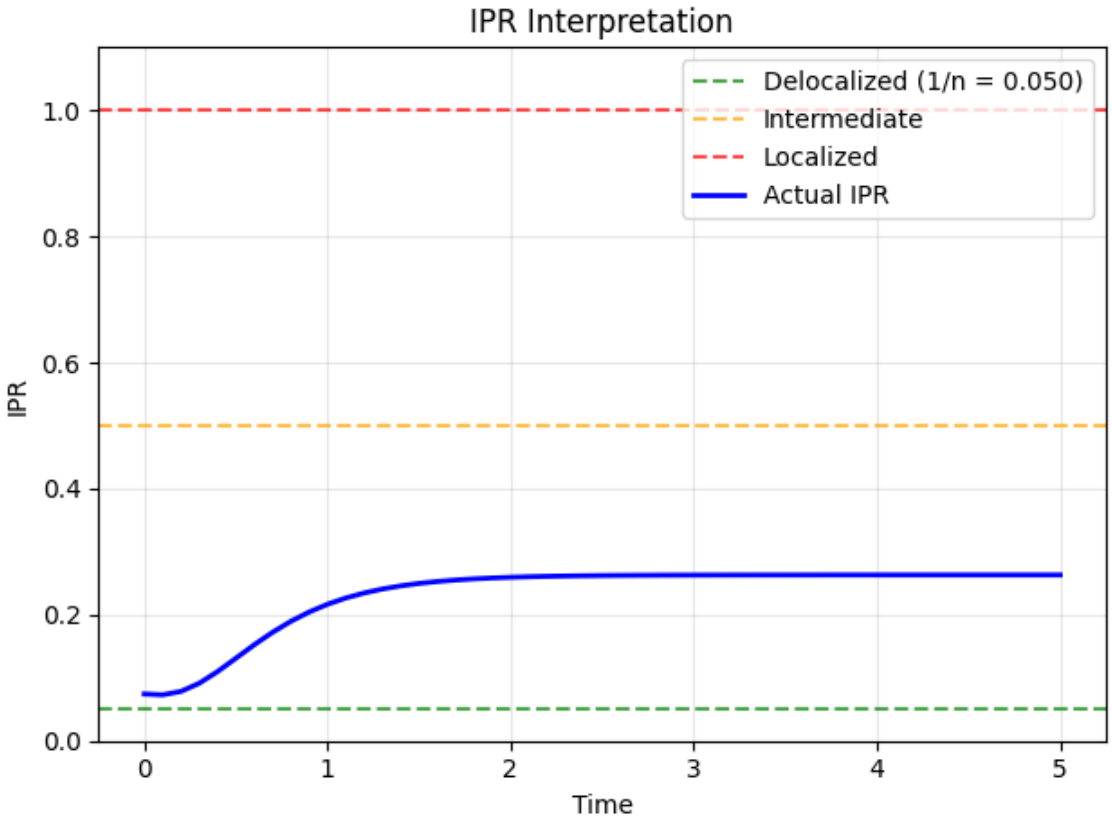

Figure 2: The IPR analysis and segregation into separate localized regions based on the property of the network input for scale free network

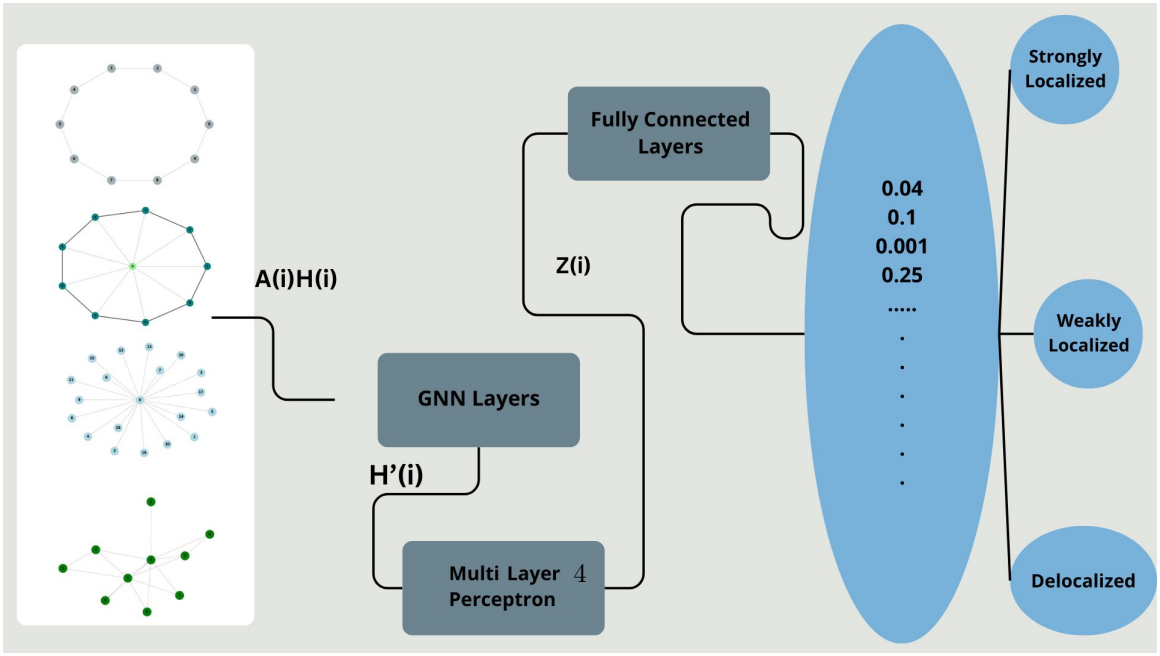

Figure 3: The General architecture of Graph Neural Networks for the regression task over the graphs in this literature

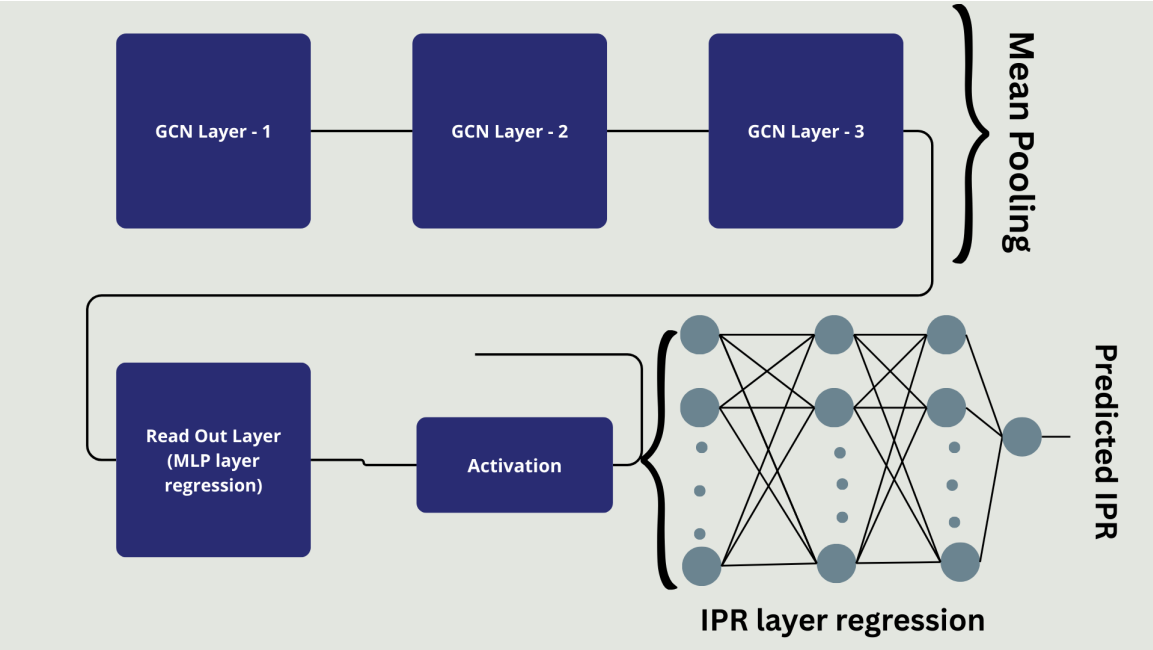

Figure 4: Architecture of the Graph Convolutional Neural Network model.Each node in the model is associated with its features.**1**The first layer of GCN is where the node aggregates and updates its features based on the intermediate neighbor and its features.**2**The second layer of GCN is where the node updates its features by aggregating the messages from neighbors to neighbors and its own. **3** The third layer of GCN aggregates the messages from neighbors to neighbors. **4** The final GCN layer performs the average pooling and then passes on the features as it is.**4** The readout layer or Multi Perceptron Layer performs the regression and classification losses are calculated through Cross Entropy Loss and finally it is passed on to **5** IPR Regression layer which performs the regression of over the IPR values and the error is predicted through the MSE loss calculating function.

III.1 GCNN Architecture

Considering a weighted graph, G with the vertex set $V = (1, ..., N)$, edge set being $E \subseteq V \times V$ composed of $|E| = M$ ordered pairs (i,j), and the weight function $W : E \to \mathbf{R}$.For each node i, define the neighborhood $N_i = (j : (j, i) \in E)$ as the set of each node connected to i and let $N_i = |N_i|$ denote the number of elements in this set.The graph G has a graph shift matrix associated with it $\mathbf{S} \in \mathbf{R}^{\mathbf{N}\times\mathbf{N}}$ whose sparsity pattern matches that of the edge set , i.e. entry $S_{ij} \neq 0$ when $(j, i) \in E$ or when $i = j$.The vertex set graph signals are $X = [x_1, x_2, ..., x_n]^T \in \mathbf{R}^{\mathbf{N}}$ in whcih component x_i is associated with node $i \in V$. $\mathbf{S}$ captures the sparsity and the locality of the relationship between components of a signal x supported on G. It is then natural to take the shift operator as the basis for defining filters for graph signals. $\phi^{(0)}$ be a $N \times N$ diagonal matrix and $\phi^{(1)}, ..., \phi^{(k)}$ be a collection of K matrices sharing the sparsity pattern $\mathbf{I_N} + \mathbf{S}$.Then for $k = 0, ..., K$ the sequence of signal $z^{(k)}$ as:

$$z^{(k)} = \Pi_{k'=0}^{k} \phi^{(k')} x = \phi^{(k:0)} x \tag{20}$$

GCNN have been quite successful in learning representations for graph data with prominent variants.All these variants written as GNN architectures in which edge varying components is fixed and given by powers of the shift operator matrix $\phi_l^{(k:0)} = S^k$

$$X_l = \sigma(\Sigma_{k=0}^{K} S^k X_{l-1} A_{lk}) \tag{21}$$

A GNN is defined by the recursive expression:-

$$x_l = \sigma(A_l(S)x_{l-1}) = \sigma(\Sigma_{k=0}^{K} \phi_l^{k:0} x_{l-1}) \tag{22}$$

here the x_0 is the input to the GNN and x_L is the output.To augment the representation power of GNNs,a multiple node feature hass to be added at each layer and this is done by defining matrices $X_l = [x_l^1, ..., x_l^{F_l}] \in \mathbf{R^{N \times F_l}}$ in which each column x_l^f represents a different graph signal at layer l.The features are cascaded through layers and are processed with graph filters and composed with pointwise nonlinearities according to :-

$$X_l = \sigma(\Sigma_{k=0}^{K} \phi_l^{k:0} X_{l-1} A_{lk}) \tag{23}$$

where $A_{lk} \in \mathbf{R^{F_{l-1} \times F_l}}$ is a parameter matrix.It is seen that the above equation represents a stack of graph filters $A_l^{fg}(S)$ applied to a set of F_{l-1} input features x_{l-1}^g to produce a set of F_l output features x_l^f.Assuming $a_{lk}^{fg} = [A_{lk}]_{fg}$ denote the (f,g)th entry of A_{lk} the above equation produces a total of $F_{l-1}F_l$ intermediate features of the form:-

$$u_l^{fg} = A_l^{fg}(S)x_{l-1}^g = \Sigma_{k=0}^{K} a_{lk}^{fg} \phi_l^{fg,(k:0)} x_{l-1}^g \tag{24}$$

A comparison between equations (21) and (24) a particular restriction yields a tensor $A(S)$ with filters of the form:-

$$A(S) = \Sigma_{k=0}^{K} a_{lk}^{fg} S^k \tag{25}$$

For some order K and scalar parameters $a_{l0}^{fg}, a_{l1}^{fg}, ..., a_{lK}^{fg}$.Simplification of the above expression by omitting layer and feature indices we have :-

$$A(S) = \Sigma_{k=0}^{K} a_k S^k \tag{26}$$

The filter above is of the form $\phi^{(0)} = a_0 I_N$ and $\phi^{k:0} = a_k S^k$ for $k \geq 1$.
The appeal for graph convolutional filters of the form (26) is due to the cause that they reduce the number of parameters from $K(M+N)+N$ of the edge varying filters to $K+1$;yielding a computational complexity of the order of $O(KM)$.While the number of parameters can be reduced in a lot of ways , the formulation in (26) is to be noted because it endows the resulting GNN with equivariance to permutation of the labels of the graph.

The **proof** for the above can be stated as follows Denoting the respective graph shift operator of the graphs G and G' as S and S'.For P being the a permutation matrix, S' and x' can be written as $S' = P^T S P$ and $x' = P^T x$ .Then, the output of the convolutional filter in (26) applied to x' is:-

$$u' = \Sigma_{k=0}^{K} a_k S'^k x' = \Sigma_{k=0}^{K} a_k (P^T S P) P^T x \tag{27}$$

By using the properties of the permutation matrix $P^k = P$ and $PP^T = I_N$, the output u' becomes :-

$$u' = P^T(\Sigma_{k=0}^{K} a_k S^k x) = P^T u \tag{28}$$

which implies the filter output operating on the permuted graph G' with input x' is simply the permutation of the convolutional filter in (26) applied to x.Subsequently since the nonlinearities of each layer are pointwise they implicitly preserve permutation equivariance; hence the output of a GCNN layer is permuted likewise.These permutations will propagate in the cascade of different layers yielding the final permuted output.

Input Layer:-
The input layer receives a normalized adjacency $(\hat{A})$ and the initial node feature $H^{(0)}$ matrices.The three convolutional layers are stacked together with an application of nonlinear activation function at the end of each layer.Each layer uses the node representation from the previous layer to compute the updated representation of the next layer.The mode of continuation of signal in GCN occurs in the following manner:-

$$H^{(l)} = \sigma(\hat{A} H^{l-1} W^{l-1}), \quad l = [1, 2, 3] \tag{29}$$

where $H^{(0)} \in \mathbf{R^{n \times d}}$ is the intial input feature matrix, and the $W^{(0)} \in \mathbf{R^{d \times k_0}}$, $W^{(1)} \in \mathbf{R^{K_0 \times k_1}}$,

$W^2 \in \mathbf{R^{k_1 \times k_2}}$ are the weight matrices for the first, second and third layers, respectively.Hence, $H^{(1)} \in \mathbf{R^{n \times k_0}}$, and $H^{(2)} \in \mathbf{R^{n \times k_2}}$ are the intermediate node features matrices and after three layers of graph convolution, final output node features are represented as $H^{(3)} \in \mathbf{R^{n \times k_3}}$The pooling function is an averaging operation performed before passing it onto the readout layer.

Readout Layer :- The averaging operation is as follows:-

$$z = \frac{1}{n}\Sigma_{j=1}^{n} h_j^{(3)} \tag{30}$$

where $h_j^{(3)} \in \mathbf{R^{1 \times k_2}}$ is the j^{th} row of $H^{(3)} \in \mathbf{R^{n \times k_2}}$ and $z \in \mathbf{R^{k_2}}$, Finally it is past onto the basic neural network in multi layer perceptron to perform regression task over graphs followed by the ipr regression layer and output the predicted IPR value as :-

$$\hat{y} = zW^{(lin)} + b \tag{31}$$

Where $W^{lin} \in \mathbf{R^{k_2 \times 1}}$ is the weight matrix and $b \in \mathbf{R}$ is the bias value,

IV Training Procedure

The derivation that follows offers and understanding of the updation of the weight matrices.The analysis is performed and with a single GCNN layer, a MLP(Multi Layer Perceptron layer) and a regression layer of fully connected neural network(Linear), the framework however explained in terms of a single GCNN layer it can be extended further for multiple layers as well.The training process has been explained via forward and backward propagation to predict the ipr value:-

Forward Propagation :-

GCN Layer :- $H^{(1,i)} = \sigma(\hat{A}H^{(0,i)}W)$

GCN Final Layer :- $z^{(i)} = \frac{1}{n_i}\Sigma_{j=1}^{n_i} h_j^{(1,i)}$

Linear layer :- $\hat{y}^{(i)} = z^{(i)}W^{(lin)} + b$

Loss Function :- $L = \frac{1}{N}\Sigma_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2$

In the above $\hat{A}^{(i)}$ is the normalized adjacency matrix.$H^{(0,i)}$ is the initial feature matrix.Further $h_j^{(1,i)} = \sigma(\Sigma_{k=1}^{n_i}\hat{A}_{jk}^{(i)}h_k^{(0,i)}W)$ is the feature vector of node j in graph i and the j^{th} row of the updated feature matrix $H^{(1,i)}$.$W^{(lin)}$ and b are the learnable weights of the linear layer.

Backward Propagation :- Inorder to compute the gradients to update weight matrices the chain rule is applied to propagate the error from output layer back through network layers.The gradient of loss with repect to the output of the linear layer is calculated as :-

$$\frac{\partial L}{\partial \hat{y}^{(i)}} = \frac{2}{N}(\hat{y}^{(i)} - y^{(i)}) \tag{32}$$

The gradients of the linear layer are being calculated and the and it is known that each graph i contributes to the overall loss L.We accumulate gradient contributions from each graph when computing gradient of loss with respect to the weight matrix $W^{(lin)}$.To obtain the gradient of the loss with repsect to the weights $W^{(lin)}$, the chain rule is applied :-

$$\frac{\partial L}{\partial W^{(lin)}} = \Sigma_{i=1}^{N}\left(\frac{\partial L}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial W^{(lin)}}\right) = \Sigma_{i=1}^{N}\frac{2}{N}(\hat{y}^{(i)} - y^{(i)})z^{(i)} \tag{33}$$

Where $\frac{\partial L}{\partial W^{(lin)}} = z^{(i)}$ from equation (31).Similarly calculating the gradient with respect to b and $z^{(i)}$ as :-

$$\frac{\partial L}{\partial b} = \Sigma_{i=1}^{N}\left(\frac{\partial L}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial b}\right) = \Sigma_{i=1}^{N}\frac{2}{N}(\hat{y}^{(i)} - y^{(i)}) \tag{34}$$

$$\frac{\partial L}{\partial z^{(i)}} = \frac{\partial L}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial z^{(i)}} = \frac{2}{N}(\hat{y}^{(i)} - y^{(i)})W^{(lin)} \tag{35}$$

Now, Calculating the gradient of the MLP Layer :-

$$\frac{\partial L}{\partial h_j^{(1,i)}} = \frac{\partial L}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial h_j^{(1,i)}} = \frac{2}{N}(\hat{y}^{(i)} - y^{(i)}) W^{(lin)} \frac{1}{n_i} \tag{36}$$

Where $z^{(i)} = \frac{1}{n_i}\Sigma_{j=1}^{n_i} h_j^{(1,i)}$ and thus $\frac{\partial z^{(i)}}{\partial h_j^{(1,i)}} = \frac{1}{n_i}$. Next on calculating gradient for the IPR Layer we have the following :-

$$\frac{\partial^2 L}{\partial W_{ipr}^{(lin)} \partial h_j^{(1,i)}} = \frac{\partial}{\partial W_{ipr}^{(lin)}} \frac{\partial L}{\partial \hat{y}^{(i)}} = \frac{\partial}{\partial \hat{y}^{(i)}} \Sigma_{i=1}^{N} \frac{2}{N} \left(\hat{y}^{(i)} - y^{(i)} \right) W^{(lin)} \frac{1}{n_i} \frac{\partial \hat{y}^{(i)}}{\partial W_{ipr}^{(lin)}} \tag{37}$$

$$\frac{\partial^2 L}{\partial W_{ipr}^{(lin)} \partial h_j^{(1,i)}} = \Sigma_{i=1}^{N} \frac{2}{N} W^{(lin)} \frac{1}{n_i} z^{(i)} = \Sigma_{i=1}^{N} \frac{2}{N} W^{(lin)} \frac{1}{n_i^2} \Sigma_{j=1}^{n_i} h_j^{(1,i)} \tag{38}$$

Finally we calculate the gradient of the GCN Layer.There are N different graphs in our dataset and each graph G_i has nodes n_i.The total gradient with respect to W accumulates the contributions from all nodes in all graphs.Hence, we sum over all nodes in each graph and then over all graphs as :-

$$\frac{\partial L}{\partial W} = \Sigma_{i=1}^{N} \Sigma_{j=1}^{n_i} \left(\frac{\partial^2 L}{\partial h_j^{(1,i)} \partial W_{ipr}^{(lin)}} \frac{\partial W_{ipr}^{(lin)}}{\partial W^{(lin)}} \frac{\partial h_j^{(1,i)}}{\partial W} \right) \tag{39}$$

The Layer output of the i^{th} graph as $H^{(1,i)} = \sigma(\hat{A}^{(i)} H^{(0,i)} W)$.Hence for a single node j in graph i, its node representation after the GCN layer is :-

$$h_j^{(1,i)} = \sigma\left(\Sigma_{k=1}^{n_i} \hat{A}_{jk}^{(i)} h_k^{(0,i)} W \right) = \sigma(q_j^{(i)}) \quad here \quad q_j^{(i)} = \Sigma_{k=1}^{n_i} \hat{A}_{jk}^{(i)} h_k^{(0,i)} W \tag{40}$$

Here , $A_{jk}^{(i)}$ refers to the element in j^{th} row of the k^{th} column of the normalized adjacency matrix, representing the connection between node j and the node k, and $h_k^{(0,i)}$ refers to the k^{th} row of the input feature matrix $H^{(0,i)}$ of graph i.
To compute $\frac{\partial h_j^{(1,i)}}{\partial W}$, we apply the chain rule as :-

$$\frac{\partial h_j^{(1,i)}}{\partial W} = \frac{\partial h_j^{(1,i)}}{\partial q_j^{(i)}} \frac{\partial q_j^{(i)}}{\partial W} \tag{41}$$

Partial derivative can be calculated with respect to $q_j^{(i)}$ as :-

$$\frac{\partial h_j^{(1,i)}}{\partial q_j^{(i)}} = \sigma'(q_j^{(i)}) \tag{42}$$

$q_j^{(i)}$ is a linear combination of the rows $H^{(0,i)}$ weighted by $\hat{A}_j^{(i)}$.In the matrix notation we can write as :-

$$\frac{\partial q_j^{(i)}}{\partial W} = \hat{A}_j^{(i)} H^{(0,i)} \tag{43}$$

where $A_j^{(i)}$ is the j^{th} row of $\hat{A}^{(i)}$.Now, we can combine the results of the chain rule and get:-

$$\frac{\partial h_j^{(1,i)}}{\partial W} = \sigma'(q_j^{(i)}) \hat{A}_j^{(i)} H^{(0,i)} \tag{44}$$

Substituting equation (38) and (44) in equation (39) we have,

$$\frac{\partial L}{\partial W} = \Sigma_{i=1}^{N} \frac{2}{N} W^{(lin)} \frac{1}{n_i} z^{(i)} = \Sigma_{i=1}^{N} \frac{2}{N} W^{(lin)} \frac{1}{n_i^2} \Sigma_{j=1}^{n_i} h_j^{(1,i)} \sigma'(q_j^{(i)}) \hat{A}_j^{(i)} H^{(0,i)} \tag{45}$$

Finally the weight matrices are updated using gradient descent as :-

$$W \longleftarrow W - \eta \frac{\partial L}{\partial W} \tag{46}$$

$$W^{(lin)} \longleftarrow W^{(lin)} - \eta \frac{\partial L}{\partial W^{(lin)}} \tag{47}$$

$$b \longleftarrow b - \eta \frac{\partial L}{\partial b} \tag{48}$$

here η is the learning rate.The above process is repeated iteratively : $forward \quad propogation \rightarrow loss \quad calculation \rightarrow backward \quad propagation \rightarrow weight \quad update$ until the model converges to the optimal set of weights that minimize the loss and from equation (45) it is clear that the minimization occurs with a rapidity of $\left(\frac{1}{n_i}\right)$.For simplicity gradient based optimization is used in the models.Adam optimization scheme was followed.

V GCNN(Training and Testing Strategy)

V.1 Training Strategy

The training process for this multi-task GNN is designed to simultaneously optimize for two distinct objectives: a classification task and a regression task. This is achieved through a combined loss function. The model is trained over a specified number of epochs, with each epoch consisting of an iterative loop over mini-batches of data from the training loader.

For each mini-batch, the model's outputs for both classification and regression are obtained. Two separate loss functions are then computed: **CrossEntropyLoss** for the classification task and **MSELoss** (Mean Squared Error) for the Inverse Participation Ratio (IPR) regression task. The total loss is calculated as the sum of these two individual losses, and this combined loss is used to drive the backpropagation and weight update process via the Adam optimizer.

To monitor the model's progress and detect potential overfitting, a validation phase is executed after each training epoch. The model is switched to evaluation mode (**model.eval()**), and its performance is assessed on a separate validation dataset. The validation losses for both tasks, along with the classification accuracy and mean absolute IPR error, are calculated and recorded. This continuous monitoring allows for a detailed view of how the model's ability to generalize to unseen data evolves throughout the training process.

V.2 Evaluation Strategy

Once the training phase is complete, the model's final performance is evaluated on a completely independent test dataset that it has never seen before. The evaluation is conducted using the **evaluate_model** function, which runs the model in inference mode without any gradient calculations to ensure the assessment is unbiased.

The evaluation strategy provides a comprehensive set of metrics to quantify the model's effectiveness on both of its tasks. For the classification task, the performance is measured using **Accuracy** and the **F1 Score**. For the IPR regression task, a more detailed analysis is performed using multiple metrics, including **Mean Abso-**

lute Error (MAE), Root Mean Square Error (RMSE), R^2 Score (coefficient of determination), and the **Pearson Correlation coefficient**. These metrics collectively provide a robust and holistic understanding of how well the model predicts the true IPR values. The final test results are then printed to the console and visualized in a series of plots, including a scatter plot of predicted vs. actual IPR values, to provide a clear and definitive benchmark of the model's performance.

VI Dataset Preparation for GCNN

VI.1 Synthetic Star Network Dataset

The model is trained and evaluated on a custom-built synthetic dataset created by the StarNetworkDataset class. The dataset simulates a star-shaped graph where a central node is connected to all other peripheral nodes. This graph structure is defined by an adjacency matrix where the center node has a strong connection to others, while peripheral nodes are not directly connected to each other.

For each data sample, the dataset generates random node features, with a specific modification to the features of the central node to create a clear class distinction. The dataset is multi-task, meaning each sample has two targets:

1. A classification label: This is a binary label determined by the mean of the features on the central node.
2. An Inverse Participation Ratio (IPR) target: This is a real-valued number calculated from the first feature dimension of the nodes. The IPR is a metric used in physics to quantify the spatial localization of a wave function and serves as the regression target for the model.

This approach creates a controlled environment for testing the model's ability to learn from the graph structure and solve both a discrete (classification) and a continuous (regression) problem simultaneously.

VI.2 Model Configuration

The GNN model used is a **LocalActivationGNN**, configured to have two distinct output branches for its two tasks. The configuration is defined in a dictionary named **model_config**.

1. **dimNodeSignals**: This parameter defines the number of features in each layer of the GNN. The architecture starts with 3 input features and progressively reduces the feature dimensions through layers of size 16, 8, and 4.
2. **nFilterTaps**: This controls the number of filter taps for each graph convolutional layer, which influences the receptive field of the filters.
3. **kHopActivation**: This specifies the number of hops (or network distance) for the activation function in each layer.
4. **dimLayersMLP**: This defines the architecture of the Multilayer Perceptron (MLP) branch used for the classification task. It consists of layers with 64 and 32 neurons.
5. **dimLayersIPR**: This similarly defines the architecture for the MLP branch dedicated to the IPR regression task, with layers of 16 and 8 neurons.
6. **GSO**: This is the Graph Shift Operator, which is the adjacency matrix of the star network. This parameter provides the model with the underlying graph structure.
7. **n_classes**: This sets the number of output classes for the classification branch.

This configuration highlights a multi-headed architecture, where the GNN's shared layers process the input data and then branch out into specialized sub-networks, each trained to solve a different task.

Figure 5: IPR Prediction for Star Network using GCNN

VI.3 Synthetic Dataset for Scale Free Network

The model's performance is tested on a synthetic dataset representing a scale-free network, a type of graph where a small number of nodes, known as "hubs," have a very large number of connections, while most other nodes have very few. The dataset is generated using a Barabási-Albert model, a common algorithm for creating such networks.

For each data sample, features are randomly generated for each node. Crucially, a distinct characteristic is introduced by significantly altering the features of the top two hub nodes, making their data values more prominent. The dataset is designed to challenge the model with two tasks:

- Classification: The binary label for each sample is determined by whether the absolute difference between the average feature values of the hub nodes and the non-hub nodes exceeds a specific threshold. This task requires the model to identify and differentiate the unique characteristics of the hub nodes within the graph structure.

- Regression: The model must predict a continuous Inverse Participation Ratio (IPR) value, which in this case is calculated based on the combined magnitude of features across all nodes. This tests the model's ability to learn and predict a global graph property from local node features.

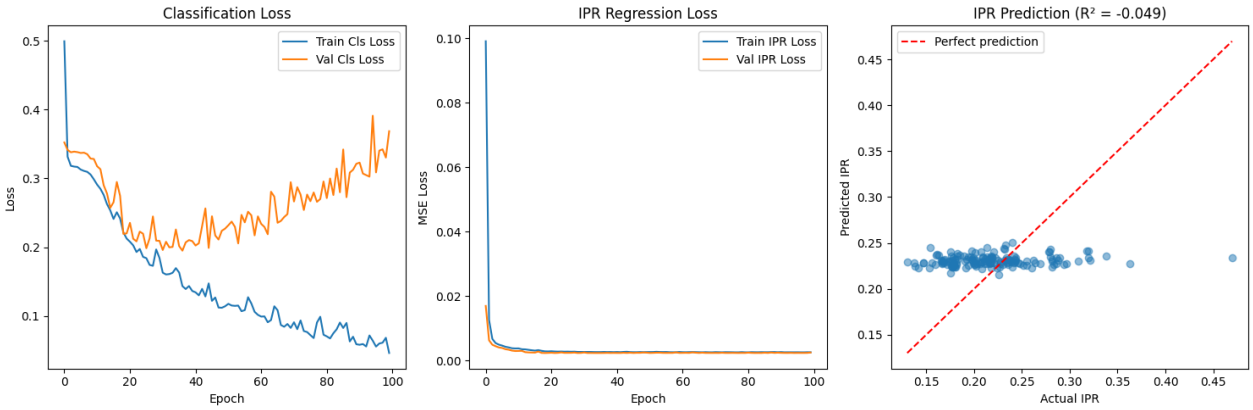

Figure 6: IPR Prediction for Scale Free Network for GCNN Model

VI.4 Synthetic Dataset for Wheel Network

The dataset is built on the wheel network topology, a specific graph structure consisting of a central hub node connected to a ring of perimeter nodes. Each perimeter node is also connected to its two immediate neighbors on the ring, forming a continuous cycle. This unique structure allows for testing the model's ability to learn from both a central, highly-connected node and a more localized, circular subgraph.

The **WheelNetworkDataset** class generates a synthetic dataset with two specific learning objectives. For each sample, the node features are initialized randomly. A specific bias is then introduced by modifying the features of the central hub node and a handful of randomly selected perimeter nodes. This feature manipulation serves as the basis for the two distinct tasks:

- Classification: A binary classification label is assigned based on a comparison between the average feature values of the central hub and the modified perimeter nodes. The model must learn to classify each sample by identifying this key difference between these two sets of nodes, highlighting its capacity to detect localized feature patterns.
- Regression: The dataset also includes a continuous regression target, the Inverse Participation Ratio (IPR). This value is calculated from the feature magnitudes of all nodes, serving as a measure of feature distribution across the entire graph. The model must predict this global property by aggregating information from all nodes and their connections.

This multi-task dataset provides a controlled environment to evaluate the model's performance on a distinctive graph topology, assessing its capacity to perform both node-level feature analysis for classification and graph-level property prediction for regression.

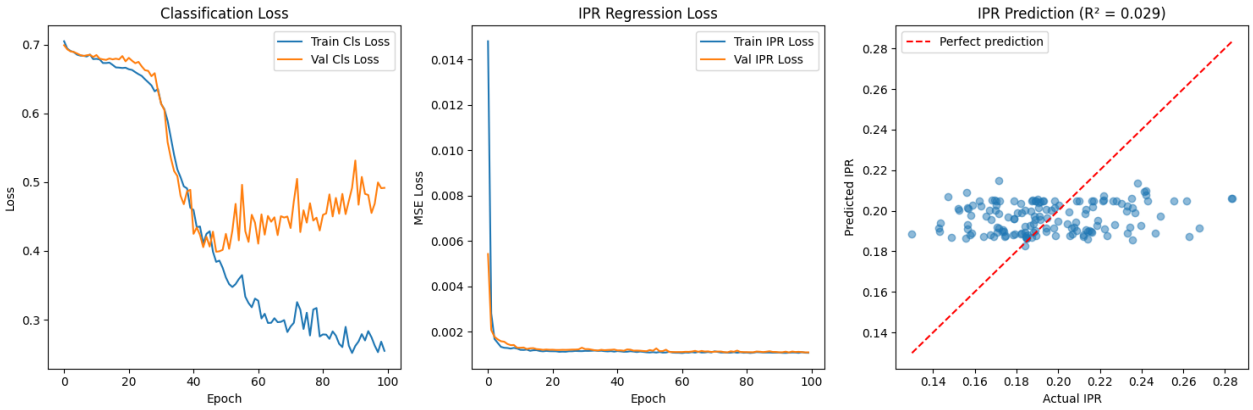

Figure 7: IPR Prediction for Whell Network for GCNN

VII ARMA Architecture

The descriptive power of the filter in equation (14) can be increased by increasing the order of K but this sets in complexity by increasing the parameters and the computational costs.It introduces numerical issues associated with higher orders of the matrix S^K.Hence setting in poor interpolatory and extrapolatory features.To counter the such effects we bring in rational rational spectral respsonse as they have better interpolatory and extrapolatory properties than polynomials.More complicated responses can be achieved in both numerator and denominator using rational functions with less learnable parameters.Auto Regressive Moving Average**(ARMA)** filters serve the purpose in such cases.

$$A(S) = \left(I + \Sigma_{p=1}^{P} a_p S^P\right)^{-1} (\Sigma_{q=1}^{Q} b_q S^q) = P^{-1}(S)Q(S) \tag{49}$$

Here $P(S) = \left(I + \Sigma_{p=1}^{P} a_p S^P\right)$ and $Q(S) = \Sigma_{q=1}^{Q} b_q S^Q$.The relationship for the input and output in the ARMA filter can be demonstrated as :-

$$\hat{u} = \left(I + \Sigma_{p=1}^{P} a_p \Lambda^p\right)^{-1} (\Sigma_{q=1}^{Q} a_q \Lambda^q)\tilde{x} \tag{50}$$

It also follows that ARMA filters are also pointwise in spectral domain charecterized by rational spectral respsonse function:-

$$a(\lambda) = \frac{\left(\Sigma_{q=0}^{Q} b_q \lambda_q\right)}{\left(1 + \Sigma_{p=1}^{P} a_p \lambda^p\right)} \tag{51}$$

The partial fraction decomposition of the rational function $a(\lambda)$ in equation (51) provides an equivalent representation of ARMA filters.Let $\gamma = [\gamma_1, \gamma_2, ..., \gamma_p]^T$ be the set of poles and $\beta =$

$[\beta_1, \beta_2, ..., \beta_p]^T$ be the number of corresponding residues and $\alpha = [\alpha_1, \alpha_2, ..., \alpha_K]^T$ be a set of direct terms and equation (51) can be rewritten as :-

$$a(\lambda) = \Sigma_{p=1}^{P} \frac{\beta_p}{\lambda - \gamma_p} + \Sigma_{k=0}^{K} \alpha_k \lambda^k \tag{52}$$

here the α, β and γ are computed from a and b.A graph filter whose spectral response is as in equation (52) is the one in which spectral response λ is replaced by the shift operator variable S.It follows that if α, β and γ are chosen to make (52) and (51)equivalent, the filter in equation (49) is equivalent to:-

$$A(S) = \Sigma_{p=1}^{P} \beta_p (S - \gamma_p I)^{-1} + \Sigma_{k=0}^{K} \alpha_k S^k \tag{53}$$

The ARMA filter training can be trained using α, β and γ instead of training a and b.
For the Jacobi implementation of the single pole filters and circumventing the matrix inverses in equation (53), each single pole filter in (53) is taken into consideration separately and implemented out with the input output relationship:-

$$u_p = \beta_p (S - \gamma_p I)^{-1} x \tag{54}$$

The above expression is equivalent to the linear equation $(S - \gamma_p I) u_p = \beta_p x$, which we can solve iteratively through jacobi recursion.This requires separating the $(S - \gamma_p I)$ into diagonal and off diagonal elements.Beginning by defining the diagonal degree matrix $D = diag(S)$:-

$$S = D + (S - D) = diag(S) + (S - diag(S)) \tag{55}$$

hence we can write $(S - \gamma_p I_N) = (D - \gamma_p I_N) + (S - D)$, which is a decomposition on diagonal terms $(D - \gamma_p I_N)$ and the off diagonal terms $(S - D)$.The jacobi iteration k for (54) is given by the recursive expression:-

$$u_{pk} = -(D - \gamma_p I_N)^{-1} [\beta_p x - (S - D) u_{p(k-1)}] \tag{56}$$

with the initialization of $u_{p0} = x$.To execute this iteration operation we can write an explicit relationship between u_{pk} and x.Inorder to perform such a recursive operation we define the parameterized shift operator:-

$$R(\gamma_p) = -(D - \gamma_p I_N)^{-1} (S - D) \tag{57}$$

and using the above we can write the Kth iterate of the jacobi recursion as :-

$$u_{pk} = \beta_p \Sigma_{k=0}^{K-1} R^k(\gamma_p) x + R^K(\gamma_p) x \tag{58}$$

Signal u_{pk} in (58) for a convergent jacobi recursion relation converges to the output u_p of the single pole filter in (54).Truncating (58) at a finite K yields an approximation in which single-pole filters are written as polynomials on the shift operator $R(\gamma_p)$, single pole filter is approximated as a convolutional filter of order K in which shift operator of graph S is replaced by the shift operator $R(\gamma_p)$ defined in (57).This convolutional filter uses parameters β_p for $k = 0, ..., K - 1$ and 1 for $k = K$.
For using Jacobi iterations to approximate all single pole filters in (53) and that we truncate all of these iterations at K, we can write ARMA filters as :-

$$A(S) = \Sigma_{p=1}^{P} H_K(R(\gamma_p)) + \Sigma_{k=0}^{K} \alpha_k S^k \tag{59}$$

where $H_K(R_{\gamma_p})$ is a K order jacobi approximation of the ARMA filter, which as per (58) is given by:-

$$H_K(R(\gamma_p)) = \beta_p \Sigma_{k=0}^{K-1} R^k(\gamma_p) + R^K(\gamma_p) \tag{60}$$

An ARMA GNN has $(2P+K+1)F^2$ parameters per layer and a computational complexity of order $O(F^2P(MK+N))$.This decomposes as $O(PN)$ to invert the diagonal matrices $(D-\gamma_p I_N)$;$O(PM)$ to scale the non zeros of $(S-D)$ by the inverse diagonal;$O(PKM)$ to obtain the outputs of jacobi ARMA filters (59) of order K.

VIII Training and Testing Strategy for ARMA GNN

The training and evaluation process for the ARMA GNN model is designed to handle a multi-task learning problem, simultaneously optimizing for both a node classification task and an Inverse Participation Ratio (IPR) regression task. The training function orchestrates the training loop, which is divided into distinct training and validation phases for each epoch. During the training phase, the model computes a forward pass to generate both a classification output and an IPR prediction. Two separate loss functions—Cross-Entropy Loss for classification and Mean Squared Error for regression—are calculated. These are then combined into a single, weighted total loss using configurable classification weight and ipr weight parameters, allowing for a flexible balance between the two objectives. This total loss is back propagated to update the model's parameters. The validation phase, which runs without backpropagation, is crucial for monitoring the model's generalization performance on unseen data. It tracks validation loss for both tasks and measures classification accuracy and Mean Absolute Error (MAE) for the IPR prediction, providing key indicators for potential overfitting.

Following the training, the evaluation function provides a final, unbiased assessment of the model's performance on a dedicated test set. The model is set to evaluation mode to ensure consistent predictions. The function processes the test data and collects all predictions and actual values for both tasks. It then computes a comprehensive set of performance metrics. For the classification task, it reports standard metrics like accuracy, F1 score, precision, and recall. For the IPR regression task, it provides a full suite of metrics including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared (R 2) score, and Pearson correlation. By returning a single, detailed dictionary, this function facilitates a thorough analysis of the model's success in mastering both the categorical classification and the continuous regression challenges.

IX Synthetic Dataset Preparation for ARMA

IX.1 Star Network Dataset

The training and evaluation of the ARMA GNN model are conducted on a synthetic dataset representing a star network topology, a simple graph structure with a central node connected to all other peripheral nodes. This dataset is generated by the StarNetworkDataset class, which creates samples with two distinct learning objectives. For the classification task, a binary label is assigned based on a significant feature perturbation applied exclusively to the central node, challenging the model to recognize and distinguish the hub's unique characteristics. For the regression task, each sample is also associated with a continuous Inverse Participation Ratio (IPR) value, a measure of feature distribution across the graph, which tests the model's ability to predict a global graph property. The model itself is configured as an ARMA GNN, which leverages parameters such as dimNodeSignals, nDenominatorTaps, and nResidueTaps to control the depth and behavior of its graph-filtering layers. It receives the static adjacency matrix of the star network as its Graph Shift Operator (GSO). Finally,

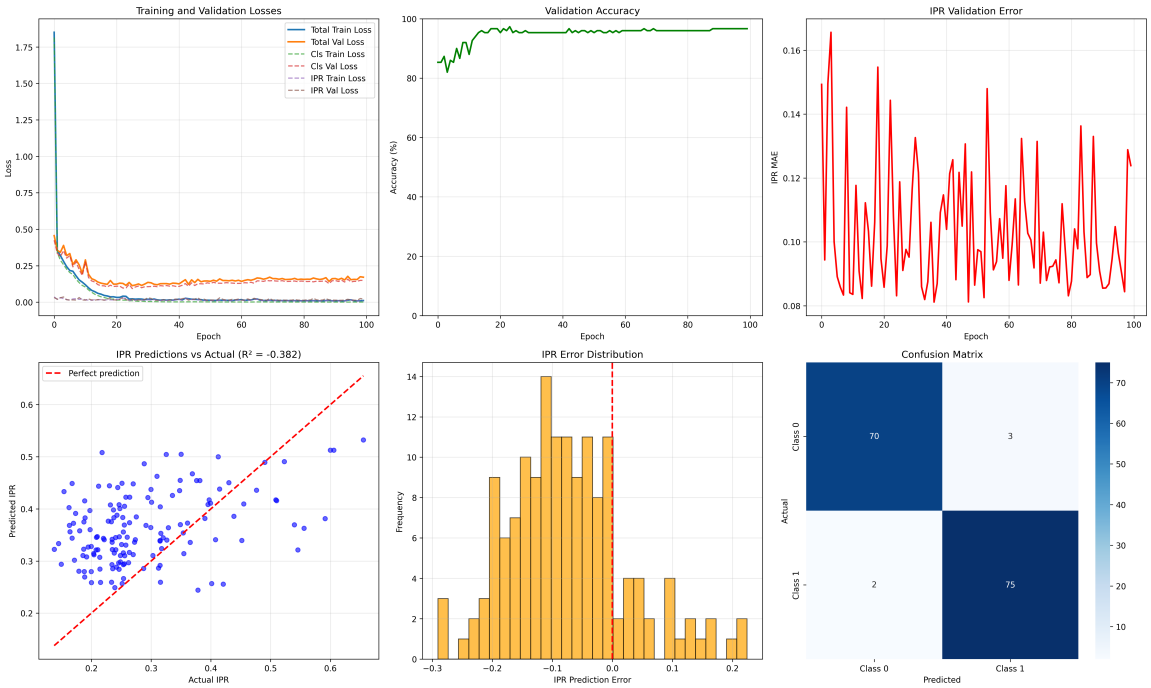

Figure 8: Star Network IPR Prediction for ARMA GNN

the model uses distinct Multi-Layer Perceptrons (MLP) in its final layers—configured by dimLayersMLP for the classification head and dimLayerIPR for the regression head—to process the learned graph representations and output both the categorical and continuous predictions simultaneously.

IX.2 Wheel Network Dataset

The provided code defines a synthetic dataset for a **wheel network**, a specific graph topology characterized by a central hub node connected to all other nodes, which themselves form a circular rim. This dataset is meticulously prepared to train a model on a multi-task learning problem, encompassing both a classification and a regression task.

For each data sample, features are randomly generated for all nodes. To introduce a key distinction, the features of the central hub node are significantly perturbed by adding a value drawn from a normal distribution. Subsequently, a small, random subset of the perimeter nodes on the rim also has its features modified. The binary label for the classification task is then determined by comparing the average feature value of the hub with the average feature value of the selected rim nodes; if the absolute difference between these two averages exceeds a threshold of 2.0, the label is set to '1', otherwise it is '0'. This forces the model to learn to distinguish the unique properties of the hub from those of the rim. Simultaneously, a continuous **Inverse Participation Ratio (IPR)** value is calculated for each graph based on its feature magnitudes. The IPR, a measure of feature distribution, is defined by the formula:

$$IPR = \frac{\sum_i |x_i|^4}{(\sum_i |x_i|^2)^2}$$

where x_i represents the features of node i. This IPR value serves as the target for the regression task, challenging the model's ability to predict a global graph property from local node features.

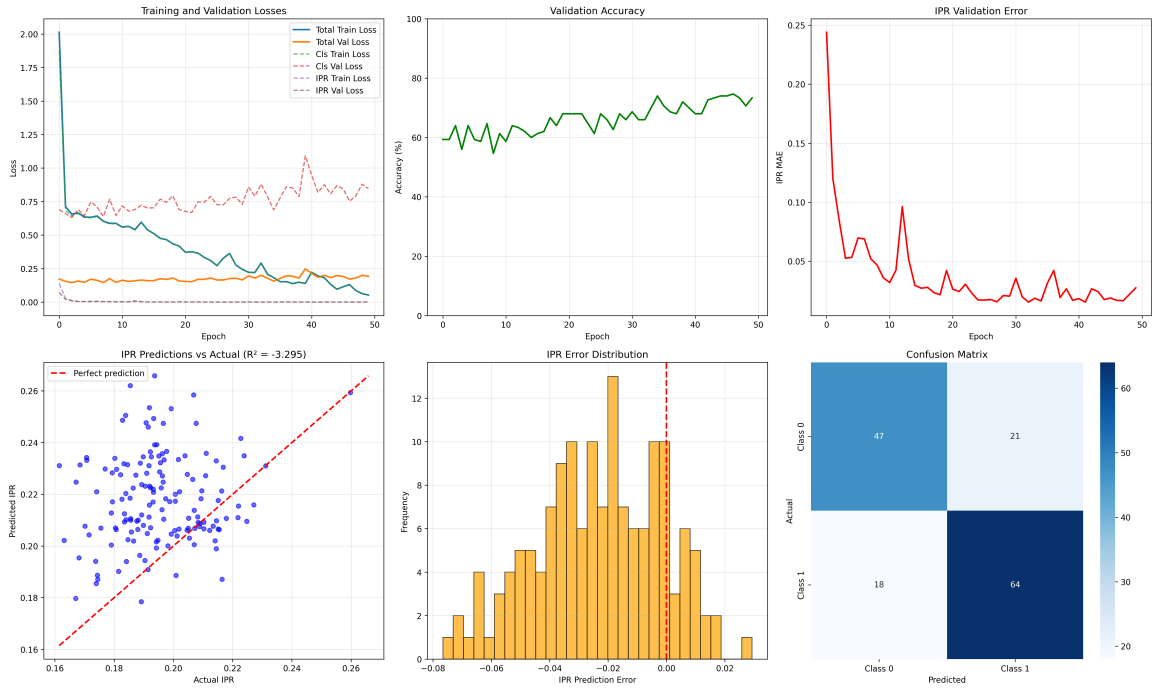

Figure 9: Wheel Network IPR prediction for ARMA model

IX.3 Scale Free Network for IPR Prediction for ARMA GNN

synthetic dataset for a scale-free network, a specific type of graph where the distribution of node degrees follows a power law, meaning a few nodes (hubs) have many connections while most have very few. The network topology itself is generated using the Barabási-Albert model, a standard algorithm for creating such structures. This dataset is meticulously prepared for a multi-task learning problem, encompassing both a classification and a regression task.

For each data sample, features are randomly generated for all nodes. To create a key challenge, the features of the top two most connected nodes (the hubs) are made intentionally distinct by adding a significant, normally distributed perturbation. The binary classification label for each sample is determined by a simple rule: a label of 1 is assigned if the absolute difference between the mean features of the hub nodes and the non-hub nodes exceeds a threshold; otherwise, the label is 0. This forces the model to learn to identify and leverage the unique characteristics of the hubs to make an accurate prediction. In parallel, the model is also tasked with a regression problem. It must predict a continuous Inverse Participation Ratio (IPR) value for each graph. The IPR is a metric calculated from the feature magnitudes of all nodes, quantifying how localized or distributed the features are across the network.

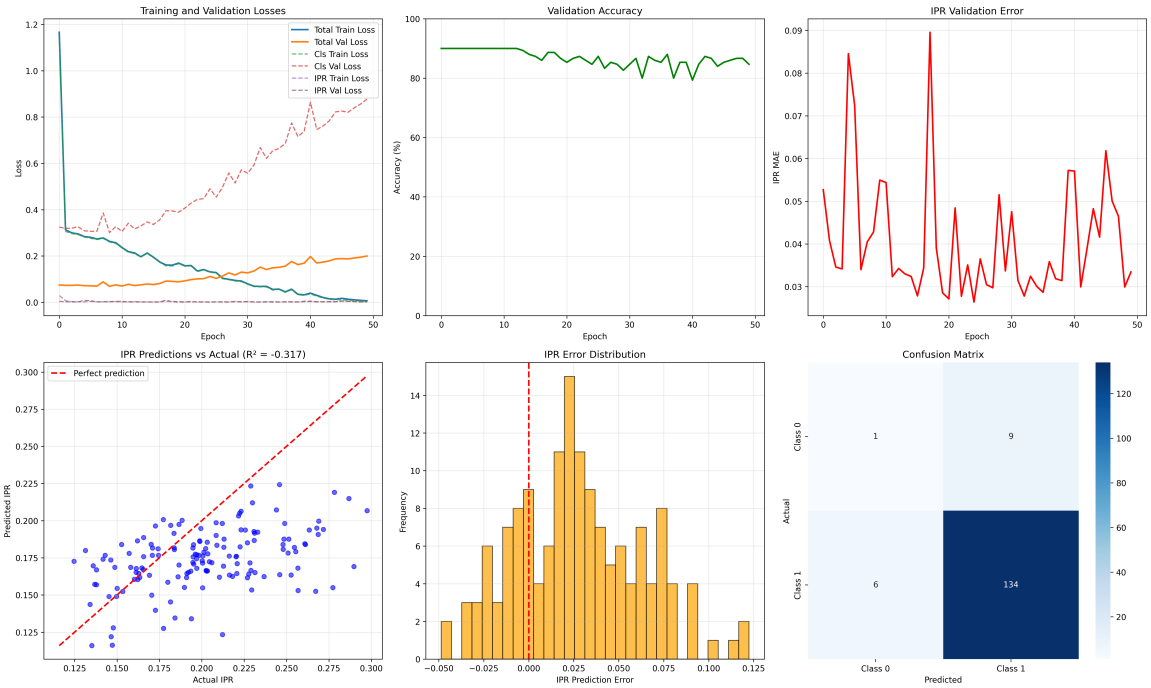

Figure 10: Scale Free Netowrk for IPR prediction for ARMA GNN

X GCAT Architecture

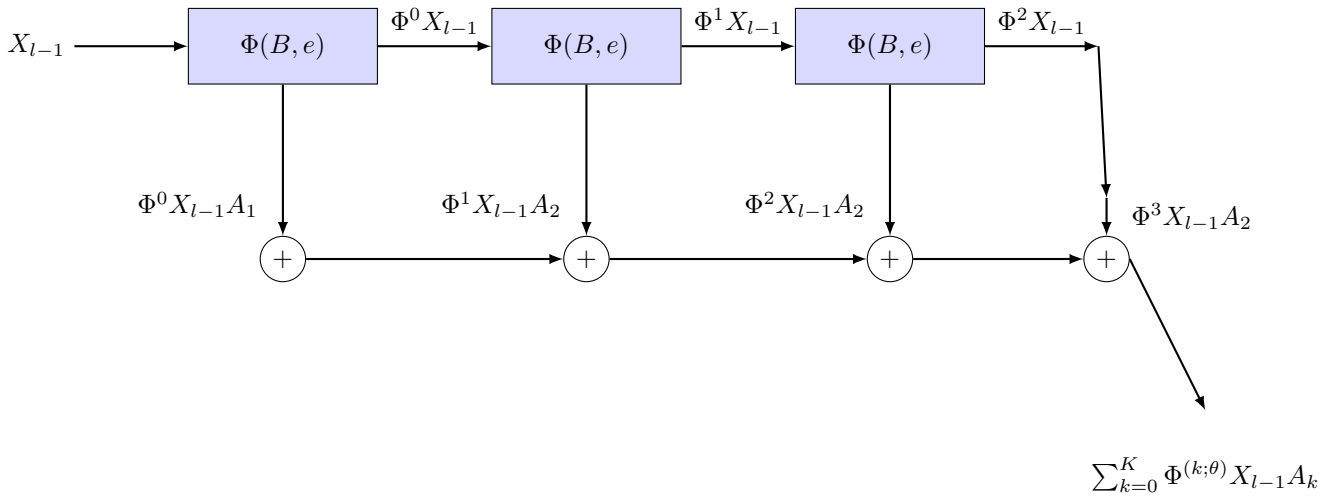

Figure 11: Higher order graph Attention filter :- Graph Convolutional Attention Filter.The input features X_{i-1} are shifted by the same edge varying shift operator $\phi(B,e)$ and weighted by the differential parameter matrices A_k.The edge varying parameters in all $\phi(B,e)$
are parameterized by the same matrix B and vector e following the attention mechanism.

The **GCNN**(Graph Convolutional Neural Networks) parameterize the **EdgeNet** by using fized underlying shift operator S and only learning the filter parameters.This shift operator often estimated from data disjointly from the GNN task, or its sepecific weights may be unknown.And graph attention mechanism comes in handy in these cases where we can parameterize the **EdgeNet** in such a way that one can learn both shift operator weights and the convolutional filter parameters.Here enters the **GCAT**(Graph Convolutional Attention Network) which uses the filter in (21) but they are convolutional in a layer specific matrix $\phi_l = \phi$ which may be different from the shift operator S.

$$X_l = \sigma\left(\Sigma_{k=0}^{K} \phi^k X_{l-1} A_k\right), here \quad \phi = \phi_l \quad and A_k = A_{lk} \tag{61}$$

Both the parameters mentioned above are layer specific.Matrix ϕ shares the same sparsity pattern as of S.The above equation defines a GNN same as defined in (23).Matrix ϕ is learned from the features X_{l-1} passed from layer $l-1$ following the attention mechanism.A trainable matrix is considered $B \in \mathbf{R}^{\mathbf{F_{l-1} \times F_l}}$ and vector $e \in \mathbf{R}^{\mathbf{2F_l}}$, and then computing the edge scores as :-

$$\alpha_{ij} = \sigma\left(e^T \left[[X_{l-1}B]_i, [X_{l-1}B]_j\right]^T\right) \tag{62}$$

for all edges $i,j \in E$.In the above equation we begin with vector with features X_{l-1} and mix them as per the parameters in the training matrix B producing a collection of graph signals $X_{l-1}B$ in which each node i has F_i features that correspond to the ith row $[X_{l-1}B]_i$ of the product matrix $X_{l-1}B$.The features of node j are concatenated with those of the features of node i producing the resulting vector of $2F_l$ components which is multiplied with the vector e.Thhe product produces the scores α_{ij} after passing through the nonlinearity $\sigma(...)$.$B = B_l$ and $e = e_l$ are global parameters for

all scores $\alpha_{ij} = \alpha_{lij}$ and are dependent on the layer index l.The score α_{ij} can be directly passed as an entry for the matrix ϕ but to promote the attention sparsity we pass the score α_{ij} through a local soft maximum operator as :-

$$\phi_{ij} = \frac{e^{\alpha_{ij}}}{\left(\Sigma_{j' \in N_i \cup i} e^{\alpha_{ij'}}\right)} \tag{63}$$

The soft maximum assigns the the edge weights ϕ_{ij} close to 1 to the largest of the edge scores α_{ij} and weights ϕ_{ij} close to 0 for the rest of the edge score α_{ij}.
The computation scores for the α_{ij} depends on the $F_{l-1} \times F_l$ parameters in B and $2F_l$ parameters in e.For the GAT in equation (61) the number of learnable parameters is at most $F^2 + 2F + F^2(K+1)$, which independent of the number of edges.The graph sparsity is obeyed throughout by ϕ and the computational complexity of (61) and its parameterization is of the order of $O(F(NF + KM))$ lesser compared to ARMA GNN.

XI Training and Testing Strategy of GCAT

The training and evaluation framework for the GCAT GNN employs a sophisticated multi-task learning approach that simultaneously optimizes both classification accuracy and Inverse Participation Ratio (IPR) regression. During training, the model processes graph-structured data through a dual-objective loss function that combines cross-entropy loss for classification tasks with mean squared error loss for IPR prediction, allowing adjustable weighting between these complementary objectives. The validation phase meticulously tracks the interplay between classification performance and localization accuracy, providing real-time feedback on model convergence and generalization capabilities. The comprehensive evaluation module delivers a holistic assessment of model performance by calculating both standard classification metrics (accuracy, F1-score, precision, recall) and specialized regression metrics (MAE, RMSE, R^2) for IPR prediction, enabling researchers to analyze the intricate relationship between graph topological properties and signal localization behavior across diverse datasets and application domains.

XI Synthetic Dataset Generation

XI.1 Star Network Datset for GCAT

The StarNetworkDataset serves as a sophisticated synthetic benchmark for evaluating graph neural networks, featuring a star topology where a central node connects to all peripheral nodes with weighted edges while maintaining self-loop connections. This dataset generates multi-dimensional node features with intentionally amplified central node characteristics to create clearly separable classes, alongside calculated Inverse Participation Ratio (IPR) values that quantify signal localization patterns across the graph structure. The comprehensive execution block orchestrates a complete machine learning pipeline, initializing the Graph Convolutional Attentional Network with optimized architectural parameters, implementing a dual-objective training regimen that balances classification accuracy with IPR regression performance, and conducting thorough evaluation with detailed metric reporting and visualization. This integrated framework produces publication-ready results including loss curves, accuracy trends, IPR prediction scatter plots, error distributions, and confusion matrices, providing researchers with both quantitative performance measures and qualitative insights into the model's ability to capture complex graph topological properties and signal localization behavior.

XI.2 Scale Free Network Dataset

The ScaleFreeDataset represents a sophisticated synthetic benchmark specifically designed to evaluate graph neural networks on complex real-world network topologies. Unlike regular or star networks, this dataset employs a scale-free network structure characterized by a power-law degree distribution, where a few highly connected hub nodes coexist with many sparsely connected peripheral nodes—mimicking the fundamental architecture of social networks, biological systems, and technological infrastructures. The dataset generation process carefully amplifies feature values around identified hub nodes while maintaining random characteristics in peripheral regions, creating a natural classification task based on the statistical disparity between hub and non-hub node features. Each sample incorporates calculated Inverse

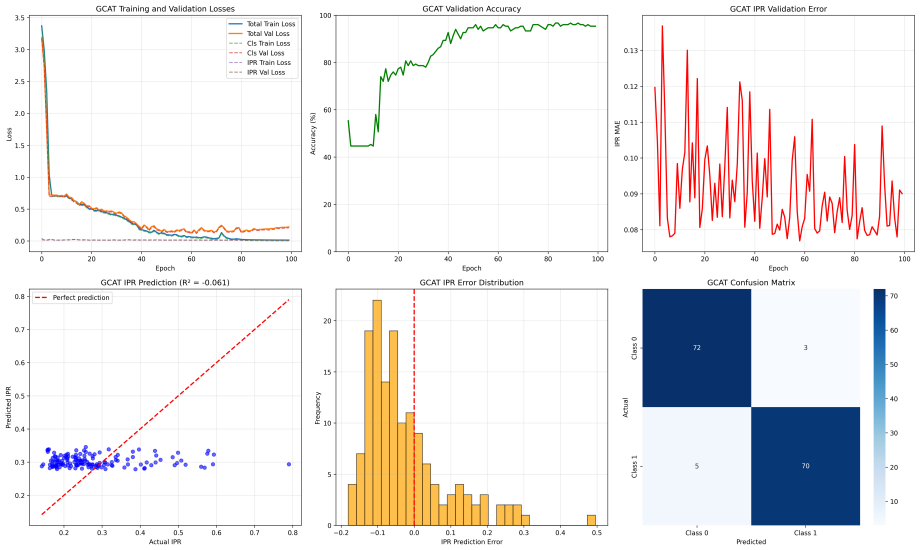

Figure 1: Star Network for IPR prediction in GCAT

Participation Ratio values that capture the localization behavior of signals across the heterogeneous network structure, providing a dual learning objective that tests models' abilities to simultaneously recognize topological patterns and quantify signal concentration. This dataset specifically challenges graph learning algorithms to distinguish between global network properties emerging from local connection patterns and localized signal behaviors that vary across the hierarchy of node connectivity, making it an ideal testbed for evaluating how well neural architectures can capture the intricate interplay between network topology and signal dynamics in complex systems.

XI.3 Wheel Network Dataset

The WheelNetworkDataset implements a specialized synthetic benchmark that captures the distinctive topology of wheel graphs, where a central hub node connects to all peripheral nodes forming a complete rim while the peripheral nodes remain unconnected to each other. This dataset generation process carefully engineers node features to emphasize the structural dichotomy between the central hub and the surrounding rim: the hub node receives consistently amplified feature values, while a strategically selected subset of rim nodes receives either strongly positive or negative feature modifications, creating a clear spectral separation between cen-

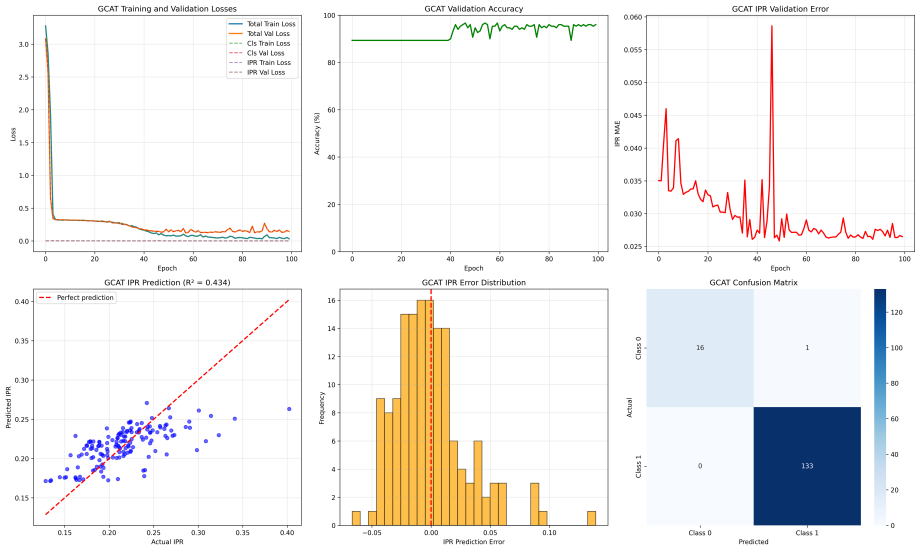

Figure 2: Scale Free Network for IPR prediction in GCAT

ter and periphery. The classification task is explicitly designed to test a model's ability to detect the feature disparity between the central hub and the modified rim nodes, with labels determined by whether the absolute difference between hub and rim feature means exceeds a defined threshold. Simultaneously, each sample incorporates calculated Inverse Participation Ratio values that quantify how signal energy distributes across the wheel's unique topology, particularly testing whether models can recognize the characteristically different localization patterns that emerge between the highly connected hub and the sparsely connected rim nodes. This dual-objective dataset specifically challenges graph neural networks to leverage both the explicit connectivity patterns of wheel graphs and the implicit feature relationships that arise from this distinctive topology, making it particularly valuable for evaluating architectural performance on systems with clear core-periphery structures, such as transportation networks, star-topology computer networks, and certain biological neural structures.

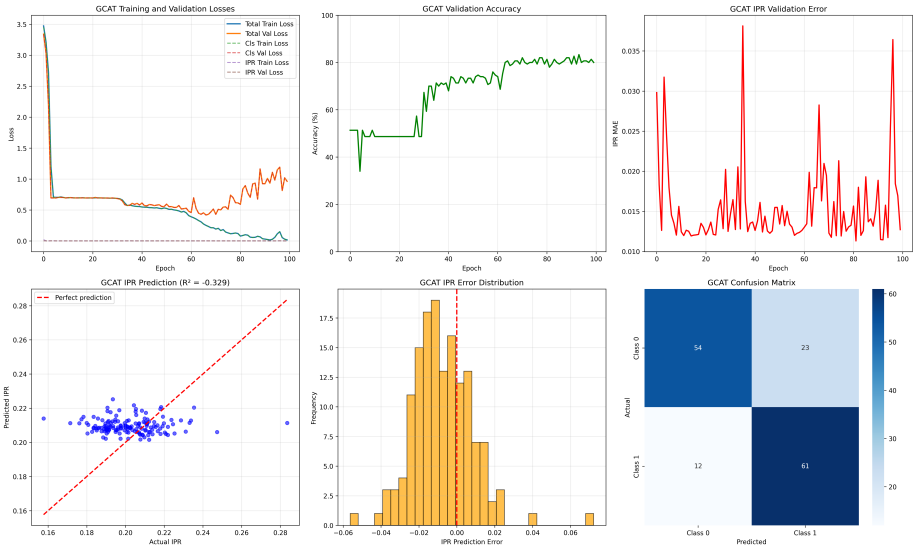

Figure 3: Wheel Network Dataset Preparation for IPR prediction in GCAT