4.1 Network Architecture In Detail

Our network architecture for link prediction on graph structured data is composed by the `GatModelV1` class and it is built around a custom `DeepGATBlockV1` module. The complete architecture is designed to process graph data, and performs node embeddings, multi-layered graph attention, and link prediction through a dot product decoder.

Overall Model Architecture

The `GatModelV1` model consists of the following components:

1. An input dropout layer to regularize input features.
2. An input linear layer, L_{in}, that projects the input features into the hidden feature space.
3. A `DeepGATBlockV1`, which applies multiple levels of GAT layers, dropout, normalization, and Feed-Forward layers, as we will discuss in next section.
4. An output linear layer, L_{out}, which transforms the processed hidden features into the final embedding space.

Mathematical Representation: The encoding process can be mathematically expressed as follows:

1. Given an input node feature matrix $X \in \mathbb{R}^{N \times F_{in}}$, where N is the number of nodes and F_{in} is the number of input features.
2. The input features undergo a linear transformation and dropout: $X' = Dropout(L_{in}(X))$, where $L_{in} : \mathbb{R}^{F_{in}} \to \mathbb{R}^{H}$, and H is the number of hidden channels.
3. The transformed features, X', are processed by the `DeepGATBlockV1`: $Z = \text{DeepGATBlockV1}(X', E)$, where E represents the edge index of the graph.
4. Finally, a linear projection is applied to the GAT block output: $Z_{out} = L_{out}(Z)$, where $L_{out} : \mathbb{R}^{H} \to \mathbb{R}^{F_{out}}$ and F_{out} is the number of output channels.

The decoder then computes the logits of edge existence based on the dot product of node embeddings extracted by the encoder: $\text{logits}_{ij} = (z_i \cdot z_j)$, where z_i and z_j are the embeddings of node i and j, respectively.

`DeepGATBlockV1` Architecture

The `DeepGATBlockV1` is a modular block composed of a series of *levels*, each of them processing node embeddings using a combination of graph attention mechanism, feed-forward network and normalization technique. Each level consists of the following operations:

1. A multi-head graph attention layer, GAT, which attends to the features of neighboring nodes, implemented with `torch_geometric.nn.GATConv` module. The output of the multi-head attention is not concatenated but rather averaged to match the input dimension. Given a node feature matrix $X \in \mathbb{R}^{N \times H}$ and an edge index E, the GAT layer calculates node representation according to the Graph Attention mechanism: $X_{att} = GAT(X, E)$, where the result X_{att} is in $\mathbb{R}^{N \times H}$.

2. An attention dropout layer to regularize attention output.

3. A residual connection between input features and the output of the graph attention layer, followed by a layer normalization. The norm operation can be expressed as: $X_1 = LayerNorm(X + Dropout(X_{att}))$.

4. A two-layer feed-forward network, FFN, each of which apply a linear transformation, non-linear ReLU activation (in between linear transformation), and dropout layer on the output. Given an input features X_1, the network can be formalized as: $X_{ff1} = ReLU(L_1(X_1))$, and $X_{ff2} = Dropout(L_2(X_{ff1}))$. Where $L_1 : \mathbb{R}^H \to \mathbb{R}^H$ and $L_2 : \mathbb{R}^H \to \mathbb{R}^H$ represent the linear transformation in the two layer FFN block.

5. A residual connection between the input features of feed-forward networks, X_1 and it's output, X_{ff2}, followed by a layer normalization. The norm operation can be expressed as: $X_{out} = LayerNorm(X_1 + X_{ff2})$.

The final output of the `DeepGATBlockV1` is further processed by a dropout to regularize it.

Mathematical Representation: Given a number of levels L:

1. for each level $l = 1, \ldots, L$:
   (a) $X^l_{att} = GAT(X^{l-1}, E)$
   (b) $X^l_1 = LayerNorm(X^{l-1} + Dropout(X^l_{att}))$
   (c) $X^l_{ff1} = ReLU(L^l_1(X^l_1))$
   (d) $X^l_{ff2} = Dropout(L^l_2(X^l_{ff1}))$
   (e) $X^l = LayerNorm(X^l_1 + X^l_{ff2})$
2. Final output: $X_{out} = Dropout(X^L)$

Graphical Representation

To give a graphical perspective of the network architecture we present the following diagram: