**III. Methodology**

In this research, we outline a multi-model framework designed to improve the detection of financial fraud using a combination of **Machine Learning**, **Deep Learning**, and **Graph Neural Network (GNN)** techniques. The methodology is organized to reflect a clear progression—from data handling and model construction to training strategy and in-depth performance evaluation—ensuring clarity and reproducibility throughout.

**A. Data Preparation and Feature Engineering**

Our work begins with the **Cora citation dataset**, a widely recognized benchmark for evaluating graph-based learning methods. In this dataset, each node corresponds to a publication, and edges indicate citation relationships between them. Every node includes a high-dimensional feature vector and an associated class label. To prepare the data for training, we apply a normalization step using NormalizeFeatures() from **PyTorch Geometric**, which helps maintain consistent input scaling and accelerates model convergence.

A significant challenge in fraud detection is **class imbalance**—genuine transactions vastly outnumber fraudulent ones. To counteract this, we employ **SMOTE (Synthetic Minority Over-sampling Technique)**. This method creates synthetic examples for underrepresented classes by interpolating feature vectors between similar instances. The result is a more balanced training set that allows our models to learn more equitably across all classes.

**B. Model Design**

We examine four distinct architectures, each tailored to capture different structural and sequential characteristics in the graph data:

* **GCN (Graph Convolutional Network):** Aggregates features from a node’s neighborhood using spectral convolutions, effectively capturing local structure in the graph.
* **GAT (Graph Attention Network):** Introduces attention weights to prioritize important neighbors during feature aggregation, enhancing robustness against noisy or irrelevant nodes.
* **GraphSAGE:** Supports **inductive learning**, enabling the model to generate embeddings for previously unseen nodes by sampling and aggregating features from their neighbors.
* **LSTM + GCN Hybrid:** Merges sequential modeling with structural learning. It processes artificially repeated node feature sequences through an LSTM and combines the temporal output with GCN-derived spatial features before final classification.

All models are implemented in **PyTorch Geometric**, utilizing layers such as GCNConv, GATConv, and SAGEConv. We carefully tune key hyperparameters like hidden dimensions, dropout rates, and attention heads through iterative experimentation to optimize performance.

**C. Training Approach**

Training is performed using the **Adam optimizer** with a learning rate of 0.01, and **CrossEntropyLoss** serves as the objective function for multi-class classification. Each model is trained for **100 epochs**, with early stopping enabled based on validation loss trends to prevent overfitting.

In the hybrid architecture, we simulate sequential data by repeating node features across time steps to produce LSTM-compatible input sequences. All training routines are run on GPU-enabled environments for improved performance, and evaluation is conducted using pre-defined training, validation, and test masks provided by the dataset.

**D. Evaluation Strategy**

The framework’s performance is assessed using a range of widely accepted evaluation metrics:

* **Accuracy:** Measures the proportion of correct predictions out of total predictions.
* **Precision & Recall:** Crucial for evaluating how well the model identifies minority classes, especially in fraud detection.
* **F1-Score:** The harmonic mean of precision and recall, offering a balanced metric in imbalanced settings.
* **ROC-AUC:** Indicates how well the model distinguishes between classes across varying thresholds.
* **R² Score:** Reflects how closely predictions align with actual values. Though more common in regression tasks, we include it to gain additional insight into multi-class prediction performance.

**E. Visualization and Interpretability**

To gain deeper understanding of the model outputs and feature embeddings, we integrate various visual tools:

* **Confusion Matrices** (via heatmaps) to observe classification accuracy across classes.
* **ROC Curves** plotted per class to visualize classification performance in terms of TPR and FPR.
* **t-SNE and PCA** plots are used to project high-dimensional embeddings into 2D space, revealing class clusters and separation patterns.
* **Training Loss Curves** provide a view into learning progression and model convergence stability.

By employing this well-rounded methodological approach, we ensure that the proposed models are both performance-optimized and interpretable—two vital aspects for real-world fraud detection systems.