## Chapter 3: Research Methodology

### 3.1 Introduction

This chapter details the systematic methodology employed to develop and evaluate a real-time fault detection system for pharmaceutical HVAC installations. The research framework is designed to be rigorous and reproducible, covering every stage from initial data acquisition to the final evaluation of the proposed machine learning models. The primary focus is on leveraging the topological structure of the HVAC system through a Graph Neural Network (GNN) and comparing its performance against a suite of well-established baseline algorithms. This chapter outlines the data source, preprocessing techniques, graph construction process, model architectures, training procedures, and the evaluation metrics used to assess performance.

### 3.2 Overall Research Framework

The methodology follows a structured pipeline, as illustrated in the framework diagram below. The process begins with raw data acquisition from the facility’s Building Management System (BMS), followed by essential preprocessing steps to ensure data quality. A critical and novel step in this research is the construction of a graph that represents the physical and systemic relationships between HVAC components. This graph, along with the processed time-series data, serves as the input for the GNN. All models are trained and rigorously evaluated, with the ultimate goal of identifying the most effective model for real-time deployment.

**[Placeholder for Figure 3.1: Research Framework Flowchart]**  
\* **Description:** A flowchart diagram illustrating the complete research pipeline.  
\* It starts with a box labeled “Raw HVAC Sensor Data (BMS/SCADA)”.  
\* An arrow points to “Data Preprocessing (Cleaning, Normalization, Feature Engineering)”.  
\* From there, an arrow splits. One path goes to “Tabular Data for Baseline Models”. The other path goes to a crucial box: “Graph Construction (Define Nodes & Edges)”.  
\* The “Graph Construction” box and the “Tabular Data” box both feed into a central “Model Training & Validation” stage. This stage lists the models: “Baselines (RF, SVM, etc.)” and “Proposed GNN Model”.  
\* An arrow from training leads to “Model Evaluation” with metrics like Accuracy, F1-Score, Precision, and Recall listed.  
\* The final box is “Performance Comparison & Real-Time Deployment Strategy”.

### 3.3 Data Acquisition and Description

The dataset for this study will be sourced from an operational pharmaceutical manufacturing facility, obtained through an industry partnership. This ensures the data reflects real-world operating conditions, complexities, and fault scenarios.

* **Data Source:** The data is collected by the facility’s integrated Building Management System (BMS) or SCADA platform.
* **Parameters Monitored:** The dataset comprises time-series data from a network of sensors across multiple Air Handling Units (AHUs) and cleanrooms. Key parameters include:
  + Room Temperature (°C)
  + Relative Humidity (%RH)
  + Differential Pressure (Pa)
  + Supply and Return Air Temperature (°C)
  + Chilled Water Valve Position (%)
  + Fan Speed / Status
  + Energy Consumption (kW)
* **Data Characteristics:** The dataset is expected to cover at least one full year of operations, with a sampling frequency of every 15 minutes. This granularity is sufficient to capture system dynamics without being computationally prohibitive.
* **Fault Labeling:** Acquiring accurate fault labels is a primary challenge. The labeling process will be a multi-step effort:
  1. **Historical Logs:** Maintenance work orders and alarm logs from the BMS will be used to identify timestamps of known equipment failures or significant operational issues.
  2. **Expert Annotation:** The data will be reviewed in collaboration with the facility’s engineering team to retrospectively label periods of anomalous behavior that may indicate incipient or undocumented faults (e.g., sensor drift, actuator sluggishness).
  3. **Handling Scarcity:** Given that faults are rare events, the resulting dataset will likely be highly imbalanced. This imbalance will be explicitly addressed during the data preparation and model training phases.

### 3.4 Data Preprocessing

Raw sensor data is often noisy and contains imperfections. Therefore, a thorough preprocessing stage is essential to prepare a high-quality dataset for model training.

1. **Data Cleaning:** Missing values, a common issue due to sensor or communication failures, will be handled using time-series imputation methods like linear interpolation or last observation carried forward (LOCF). This preserves the temporal nature of the data.
2. **Outlier Removal:** Erroneous readings or extreme spikes caused by sensor malfunctions will be identified using statistical methods (e.g., Z-score or Interquartile Range) and removed or clipped to plausible physical limits.
3. **Data Normalization:** All numerical features will be scaled to a common range. **StandardScaler** (Z-score normalization) will be used, which transforms features to have a mean of 0 and a standard deviation of 1. This is critical for the performance of distance-based algorithms like SVM and for the stable training of neural networks.

### 3.5 Graph Construction for GNN

A core contribution of this thesis is the explicit modeling of the HVAC system’s topology. To apply a GNN, the sensor network must be represented as a graph G = (V, E), where V is the set of nodes and E is the set of edges.

* **Nodes (V):** Each node will represent a unique sensor in the system (e.g., Room\_117\_Temp, AHU-09\_Fan\_Speed). At any given time t, the feature vector for a node will be its preprocessed sensor reading at that time.
* **Edges (E):** Edges will be defined based on two types of physical relationships, creating a static, unweighted, and undirected graph:
  1. **System Connectivity:** Edges will connect nodes that are part of the same physical subsystem. For example, an edge will exist between the AHU-09\_Fan\_Speed sensor and the Room\_117\_Temp sensor because AHU-09 directly supplies air to Room 117. This information will be derived from HVAC schematic diagrams.
  2. **Spatial Proximity:** Edges will connect sensors in rooms that are physically adjacent. This captures thermal interactions between neighboring zones and will be determined from building floor plans.
* **Adjacency Matrix:** These relationships will be encoded into an **adjacency matrix (A)**, where A[i, j] = 1 if an edge exists between node i and node j, and 0 otherwise. This matrix is a fundamental input for the GNN model.

**[Placeholder for Figure 3.2: HVAC System Graph Representation]**  
\* **Description:** A simplified schematic.  
\* On the left, show a simple HVAC diagram (like Figure 2 from the proposal ) with an AHU serving three rooms.  
\* An arrow points to the right, showing the corresponding graph structure.  
\* The AHU is a central node. Edges connect it to three “Room” nodes.  
\* Additional edges connect adjacent Room nodes to each other, illustrating both system and spatial connectivity. Each node is labeled with its corresponding sensor (e.g., “Room 1 Temp”).

### 3.6 Model Development and Training

This research will compare the proposed GNN model against a variety of established baseline models to benchmark its performance comprehensively.

#### 3.6.1 Data Splitting

The dataset will be split chronologically into three sets to prevent data leakage and simulate a real-world scenario:  
\* **Training Set (60%):** The oldest data, used to train the models.  
\* **Validation Set (20%):** The subsequent data, used for hyperparameter tuning.  
\* **Test Set (20%):** The most recent data, held out completely until the final evaluation to provide an unbiased assessment of model performance.

#### 3.6.2 Baseline Models

A suite of classical machine learning algorithms will be implemented using the **scikit-learn** library in Python. These include:  
\* Logistic Regression (LR)  
\* Random Forest (RF)  
\* Gradient Boosting (GB)  
\* Support Vector Machine (SVM)  
\* K-Nearest Neighbors (KNN)

These models will be trained on a tabular format of the data, where each row is a timestamp and columns are the sensor features. Hyperparameter tuning for each model will be performed using Grid Search with 5-fold cross-validation on the training set.

#### 3.6.3 Proposed GNN Model

The GNN model will be implemented using the **PyTorch Geometric (PyG)** library. A Graph Convolutional Network (GCN) architecture will be designed for the node classification task (classifying the state of each sensor/node).

**[Placeholder for Figure 3.3: GCN Model Architecture]**  
\* **Description:** A diagram showing the layers of the GNN.  
\* Input layer shows the Graph G (with node features X and adjacency matrix A).  
\* This feeds into “GCN Layer 1 (with ReLU activation)”.  
\* This feeds into “GCN Layer 2 (with ReLU activation)”.  
\* The output of the GCN layers is a set of node embeddings.  
\* These embeddings go into a “Fully Connected Layer” and finally a “Softmax Activation” layer that outputs fault probabilities for each node.

The architecture will consist of:  
\* **Input Layer:** Takes the graph snapshot at each timestamp as input.  
\* **Graph Convolutional Layers:** Two GCN layers will be used to aggregate information from neighboring nodes, allowing the model to learn context from connected components. The ReLU activation function will be used to introduce non-linearity.  
\* **Output Layer:** A final linear layer followed by a Softmax activation function will output the probability for each class (e.g., ‘Normal’, ‘Fault Type A’, ‘Fault Type B’) for each node.  
\* **Loss Function:** Due to the class imbalance, a weighted Cross-Entropy Loss function will be used to penalize misclassifications of the minority (fault) classes more heavily.

### 3.7 Model Evaluation Metrics

To rigorously evaluate and compare the models, a standard set of classification metrics will be used. Given the imbalanced nature of fault data, accuracy alone is insufficient.

* **Accuracy:** The overall percentage of correct predictions.
* **Precision:** Of all the instances the model predicted as faults, what percentage were actual faults? High precision is crucial for minimizing false alarms.
* **Recall (Sensitivity):** Of all the actual faults that occurred, what percentage did the model correctly identify? High recall is critical for ensuring faults are not missed.
* **F1-Score:** The harmonic mean of Precision and Recall, providing a single score that balances the two. It is the primary metric for evaluating performance on imbalanced datasets.
* **Confusion Matrix:** A table used to visualize the performance of a classification model, showing the counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

### 3.8 Experimental Setup

All experiments will be conducted in a controlled computational environment to ensure reproducibility.  
\* **Software:** Python 3.9+, PyTorch, PyTorch Geometric, scikit-learn, Pandas, NumPy, Matplotlib.  
\* **Hardware:** A workstation equipped with a multi-core CPU and at least one NVIDIA RTX 30-series (or equivalent) GPU with >8GB of VRAM to accelerate the training of the GNN model.