**Problem Statement Discussion**

The problem statement encompasses the categorisation of electrocardiogram (ECG) signals into distinct diagnostic groups. ECG classification holds potential as a valuable tool for predicting blood pressure (BP) levels in patients. By analysing and interpreting ECG data, specific patterns and anomalies can be recognised, providing crucial insights into an individual's cardiovascular health. Furthermore, the integration of these ECG findings with BP prediction enhances our ability to detect and manage hypertension at an early stage, thereby enabling proactive healthcare interventions.

**Process and Approach**

1. To begin my first deep learning project, I found a sample MATLAB code that classifies ECG data from the PhysioNet 2017 challenge. This data consisted of single short ECG lead recordings categorised into two classes: Normal and Atrial fibrillation.
2. The MATLAB example employed a deep LSTM network for classification, which I converted into Python. Once I familiarised myself with implementing Deep Neural Networks, I searched for multi-class ECG datasets and came across the PTB-XL dataset on Physionet.
3. The PTB-XL dataset contains 21799 clinical 12-lead ECG signals classified into five diagnostic classes: ‘Normal ECG’, ‘Myocardial Infarction’, ‘ST/T Change’, ‘Conduction Disturbance’, and ‘Hypertrophy’.
4. However, the dataset was imbalanced, with around 10,000 signals classified as Normal. This imbalance could potentially bias the model towards classifying ECGs as Normal more accurately than other classes. To address this issue, I augmented the ECG signals from the minority classes by repeating the available signals and adding noise to each one. This augmentation achieved two goals: balancing the number of signals across classes and increasing the overall dataset size.
5. Initially, I experimented with simple deep CNN models consisting of 2-3 convolutional layers and fully connected layers. This approach yielded a test accuracy of 83%, but I struggled to improve it further.
6. As a result, I decided to explore the Residual Network model. Starting with the original ResNet architecture, I implemented several modifications to enhance its accuracy.
7. I achieved a training accuracy of 97% , a validation accuracy of 93%.