DETECTION & LOCALIZATION OF MYOCARDIAL INFARCTION -REPORT

OVERVIEW:

The causes of Myocardial Infarction (MI), or a heart attack, involve some blockage of one or more of the coronary arteries. The classification of electrocardiograms (ECG) plays a vital role in the clinical diagnosis of heart disease.

I have used One-Dimensional Convolutional Neural Network (CNN) and Bidirectional Long-Short Term Memory Network (LSTM) for automated detection and localization of MI ailment by extracting thirty-six morphological features from 12-lead electrocardiogram (ECG) beats (amplitude of T-wave, the amplitude of Q-wave, and ST-segment deviation for all leads). Overall Test Accuracy obtained = 95. 33% **.**

DATA:

This project work uses the PTB-XL dataset (present as open-access on Physionet). The PTB-XL ECG dataset is a large dataset of 21837 clinical 12-lead ECGs from 18885 patients of 10-second length. The records for the ECG signals are present at sampling frequencies 100Hz and 500 Hz. For this project, I have used the ECG data recorded at a 500Hz of sampling frequency. Further, 9528 records for Normal sinus rhythm (NORM) and 5486 records for MI categorized into eight subcategories as anterior MI (AMI), anterio-septal MI (ASMI), anterio-lateral MI (ALMI), inferior MI (IMI), inferio-lateral MI (ILMI), inferio-posterior MI (IPMI), inferio-posterio-lateral MI (IPLMI), posterior MI (PMI), and lateral MI (LMI) are present in the dataset. The table below shows each of these MI classes with their frequency in the PTB-XL dataset.

Classes	Frequency
AMI	290
ASMI	1883
ALMI	164
IMI	2329
ILMI	393
IPMI	30
IPLMI	50
PMI	14
LMI	132
NORM	9528

This project (as mentioned earlier) is to build a classifier algorithm to predict the type of MI or NORM from a given 12-lead ECG signal as data. The table above also shows that the dataset is highly unbalanced.

BALANCING THE DATASET:

Due to the unavailability of GPUs and extra RAM, I have considered taking only 1000 samples from each class, which shall perfectly suffice. This was done by oversampling those classes, which have a frequency lower than 1000, and undersampling the classes with a frequency higher than 1000. I used the 'imblearn's SMOTENC' class available in Python for oversampling and undersampling purposes using categorical features. So, doing this will ultimately balance the initial data. Further, each ECG record present in the dataset was categorically labeled.

DATA PRE-PROCESSING AND FEATURE EXTRACTION FROM BEATS:

Each ECG signal (at sampling frequency = 500 Hz) from the dataset was read into a matrix with shape [5000, 12] using the 'wfdb' package. Each lead/channel was extracted one-by-one from the signal and sent to get

denoised and detrended. The data-filtering function was built using the 2nd order digital Butterworth and zero-phase digital filters implemented in Python using the SciPy library, which also worked as high and low pass filters. After passing these filters, all the unwanted frequencies ranging below 0.5Hz and above 40Hz were successfully eliminated. Furthermore, each of the twelve leads of ECG signals was sent for ECG-delineation, where I used the Hamilton Segmenter peak detection algorithm to detect all the R-peaks from it. Based on those detected R-peaks, I used the inbuilt 'discrete wavelet transform' method to detect the QRS-onsets, QRS-offsets, and T-peaks, whereas the 'peak' method to detect the Q-peaks from all the beats. For an ECG beat, the primary indicators are Q wave, T wave, and ST level elevation or depression. The ST level elevation or depression is measured by seeing the level of the R offset concerning the iso-electric line or the corresponding R-onset value. These three time-domain features, i.e., T wave amplitude, Q wave amplitude, and ST deviation measure, are extracted for each beat and combined for 12-leads to form a 36-dimensional feature vector. Later, these features are used for MI detection and localization purposes.

*The number of beats of nine types of Myocardial Infarction and healthy subjects is shown in the below table. A total of **37,685 ECG beats** of healthy and nine types of MI are used in this project.*

MI Type	Number of Beats
AMI	3710
ASMI	3188
ALMI	4515
IMI	3876
ILMI	3951
IPMI	2979
IPLMI	3642
PMI	4245
LMI	4448
NORM	3131

TOTAL	37,685

TRAIN-TEST SPLIT:

Before moving forward to training a model, the overall data was split into training (75%) and testing (25%) data using the StratifiedKFold cross-validation method with shuffling allowed. The StratifiedKFold approach ensures equal distribution from all the classes.

MODEL TRAINING and RESULTS:

Since ECG signals are time-series data with time sequences, architectures consisting of LSTM and 1D CNN layers would be the best suitable models.

Convolution Layers: Used two Conv1D layers four times (total = 8), every time followed by a MaxPooling layer (MaxPooling1D) and a Dropout layer.

Since the input matrix was a column vector with shape $(36, 1)$, I chose to use a maximum number of filters for each of the convolutional layers but kept the filter size (kernel size) as small as possible to hold the compatibility of convolution operation and not reduce the size from 36 any further. So, the no. of filters were kept in the range $128 - 256$ while their sizes were held in the range $3 - 1$. These values would lead to a significant increment in the output size as compared to the input's size. Each convolutional layer was activated using the 'ReLU' activation function because compared to the 'sigmoid' and 'tanh' functions, 'ReLU' converges faster and helps get over the over-fitting problem.

Pooling Layers: Used MaxPooling in the 3rd, 7th, 11th, and 14th layers. Since we couldn't afford to decrease the input size (or shape), I kept the pool_size for each case = 1.

Dropout Layers: The dropout layers at a regular interval have been added to prevent overfitting. Checked the value of keep_prob by looping it over from 0.1 to 0.5 and tuned it for the deal, which gave the best result.

Bidirectional LSTM Layers: Tried several sequence-based layers, with the best performing being Bidirectional LSTM layers. These layers essentially extract the time-domain features from the output of the convolutional layers. The value of hidden units was kept as large as 256 to ensure a larger output size.

After this, I flattened the output layer and trained some fully connected layers with hidden units = 256(fine-tuned), providing the $L2$ regularization constraint as kernel and bias regularizers (with regularization constant = 0.01(fine-tuned)). I also used BatchNormalization() layers to standardize the input and prevent generalization errors.

In the end, the model contained the most crucial 'softmax layer' or the classification layer.

For training, fit the model and trained it for 70 epochs, keeping batch size = 128 (must be a power of 2). These hyperparameters had been checked and tuned up.

The model showed an overall accuracy of 95.33% on the test dataset.

After looking at the confusion matrix, the accuracy of the model on each of the classes are:

Class	Accuracy
ALMI	100%
IPLMI	100%
LMI	100%
PMI	100%
IPMI	99.87%
AMI	98.49%
ILMI	97.25%
IMI	86.07%

ASMI	86.19%
NORM	85.4%