ECG Heartbeat Categorization Project

Team members:
Xu Muzi (1005641)
Emmanuel J Lopez (1005407)

GitHub Repo

Project Objectives

An ECG signal is a recording of the electrical activity of the heart. It is a commonly used diagnostic tool in cardiology to detect various heart conditions such as arrhythmias, heart attacks, and other abnormalities.These signals are preprocessed and segmented, with each segment corresponding to a heartbeat.

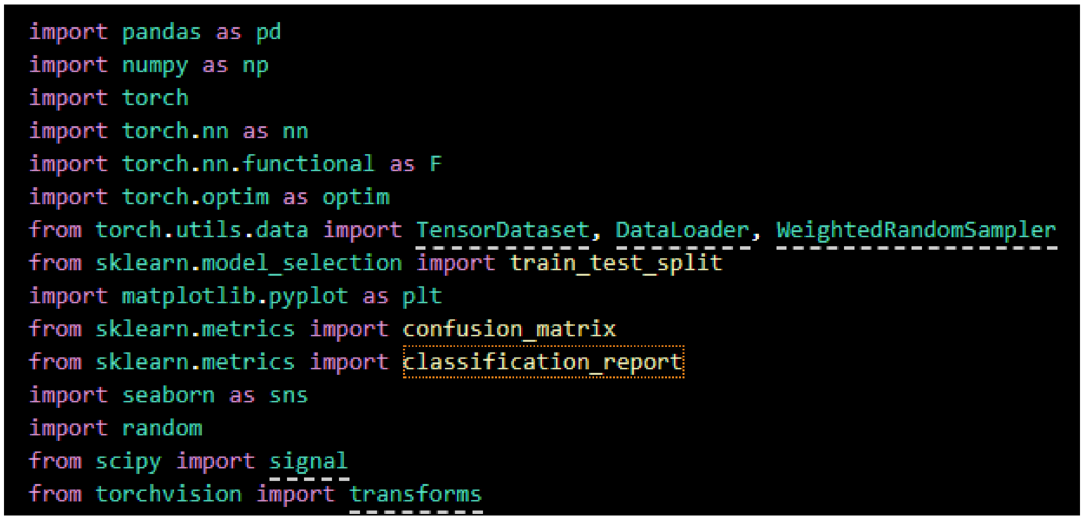
In the dataset, we are given one class of normal ECG data and four other classes of abnormal beat data, shown in the following.

- Class 0 (N): Normal beat (Non-ectopic beats)
- Class 1(S): Supraventricular ectopic beats
- Class 2 (V): Ventricular ectopic beats
- Class 3 (F): Fusion Beats
- Class 4 (Q): Unknown Beats

The goal of the project is to train a model that accurately classifies the ECG dataset into their corresponding classes and minimizes erroneous classifications.Accurate classification of ECG signals is important for early detection and timely treatment of these conditions, which can ultimately save lives.

Data Visualization

The code first begins by importing the necessary libraries needed for the data visualization, augmentation and model-building, including pandas, NumPy for data manipulation, Pytorch library for constructing the CNN model and seaborn and seaborn and matplotlib for visualization. The details of the import libraries are shown below.



```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import TensorDataset, DataLoader, WeightedRandomSampler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
import random
from scipy import signal
from torchvision import transforms
```

Figure 1. Import Libraries.

Since the dataset samples are categorized into five classes, and the shapes of the signals are what the model analyzes to determine the corresponding class the samples belong to, we decide to visualize these signals.

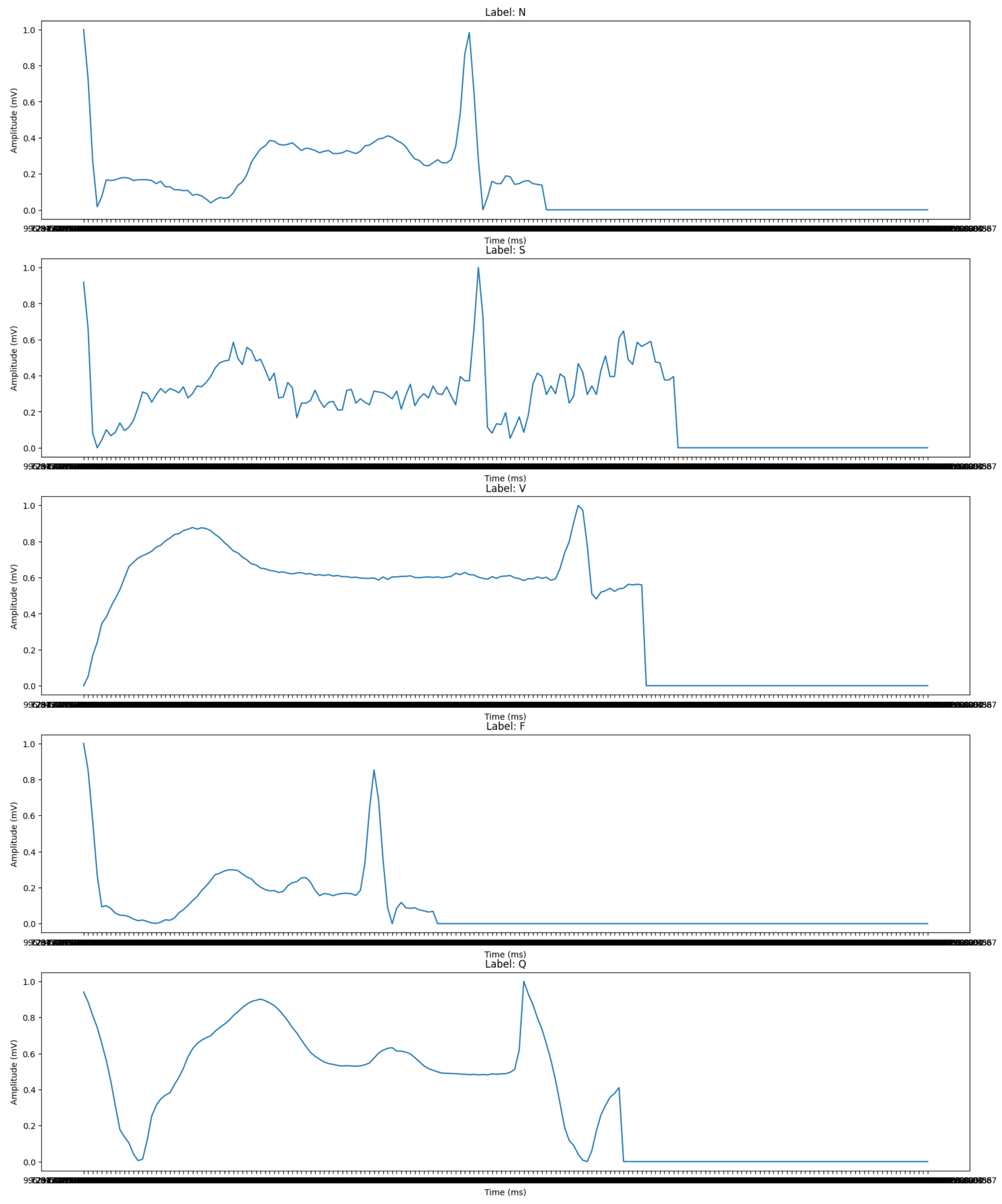


Figure 2. ECG Dataset Signal Visualization (Non-Normalized)

From the visualization, we notice that the data is not normalized. This may cause the model to have difficulties categorizing the samples into the correct classes. Therefore, we normalized the data before proceeding to build the CNN model.

The following figure shows the normalized signals.

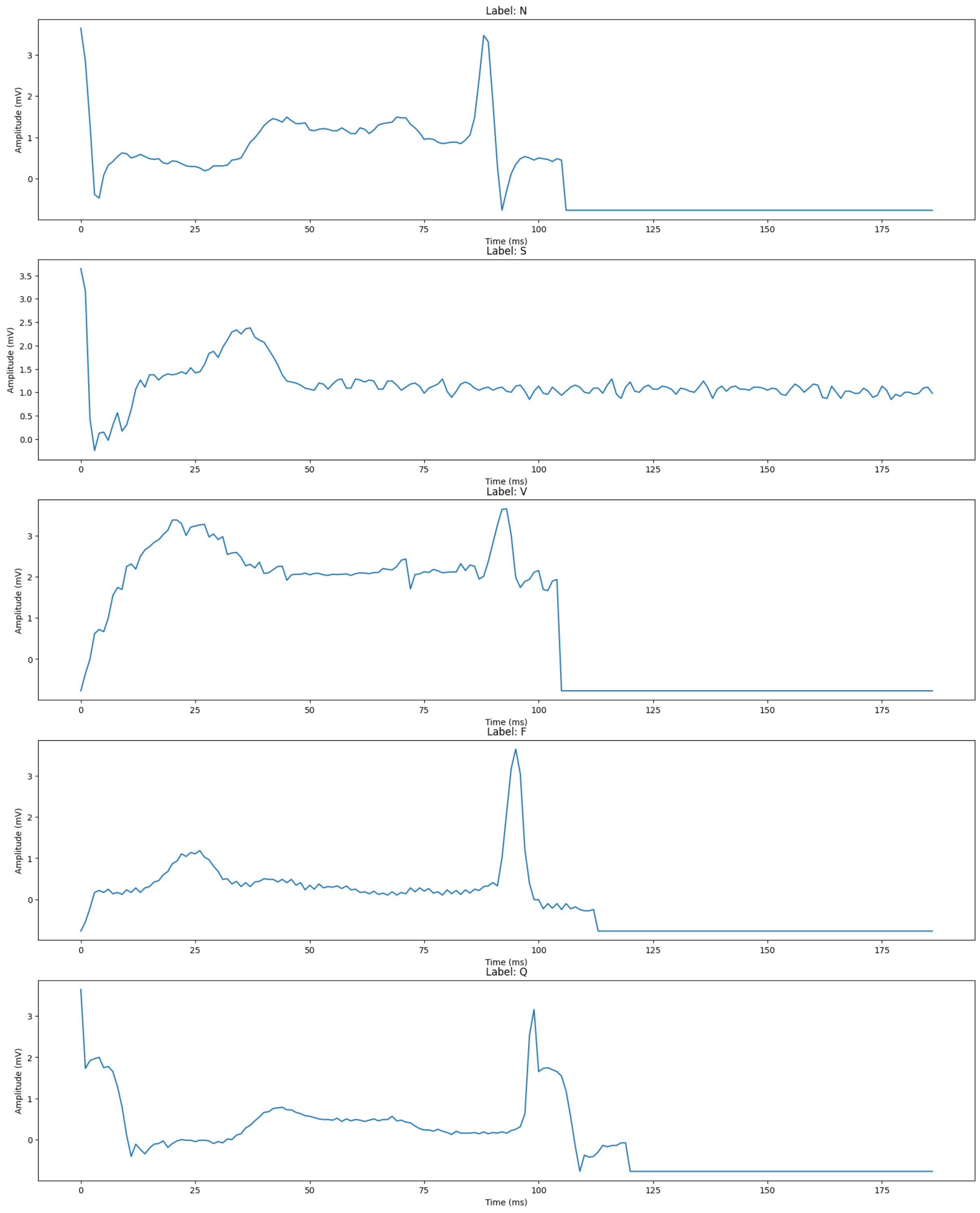


Figure 3. ECG Dataset Signal Visualization (Normalized)

Model Architecture

Considering the importance of spatial dependency for a recording of a heartbeat, we decided to use a Convolutional Neural Network for 1 dimensional data to classify the samples to their respective type of arrhythmia.

Before the model was created, the dataset's class imbalance had to be fixed. We chose to do so via data augmentation using random oversampling of the minority classes for the training data. This was done via the WeightedRandomSampler from Pytorch, and proved to be a sufficient fix for the class imbalance.

Figure 4. Classes Before Augmentation

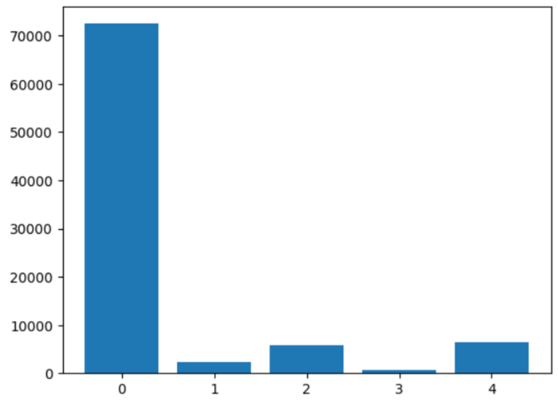
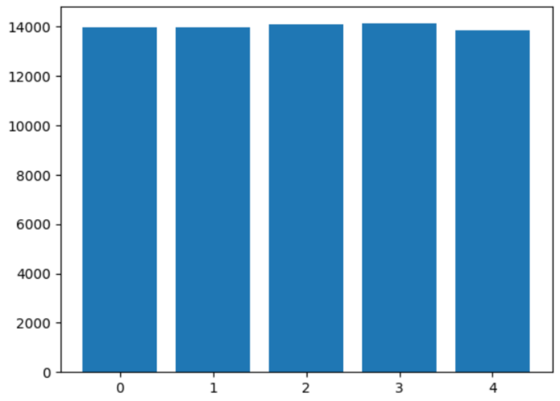
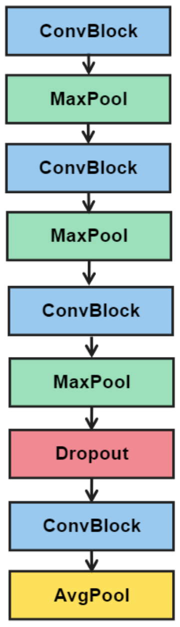


Figure 5. Classes After Augmentation



The neural network uses a custom block, consisting of 3 sequential 1D Convolutional layers and 1 Batch Normalization layer after the first convolution. An ReLU activation function is used after each convolution layer in the block.

The model itself consists of 5 blocks interlaced with max pooling layers in the order below.



Inspired by AlexNet, the convolutional blocks start with a larger kernel size of 11 to capture the larger features, and decreasing the kernel size in later layers to capture the finer details. A single dropout layer is used to aid in avoiding overfitting the model. The number of channels in each block increases up to 512 channels before decreasing in the last two layers to 5 channels. When each of the 5 images from the output channels is passed through the AdaptiveAvgPool1d layer with output size 1, it can be used for classification to the 5 output labels after flattening.

Figure 6. Model Architecture

Training And Loading Pretrained Model

To train the model from scratch the entire notebook has to be run, bar the data visualization section. To load the pre-trained model, run the **Imports and Initialization**, **Dataloading**, and **Model** sections up to the **Trainer function**, then the **Evaluation** section can be run to test the model using the pretrained weights.

Project Summary

The model we have chosen is 1D Convolutional Neural Network and it achieves an overall accuracy of **98.51%**. The CNN architecture is optimized by the Adam Optimizer, trained at the learning rate of 0.001 and at 50 epochs. We can see from the following graph that the model does not face any underfitting or overfitting issues as both the training loss and test (validation) loss converge.

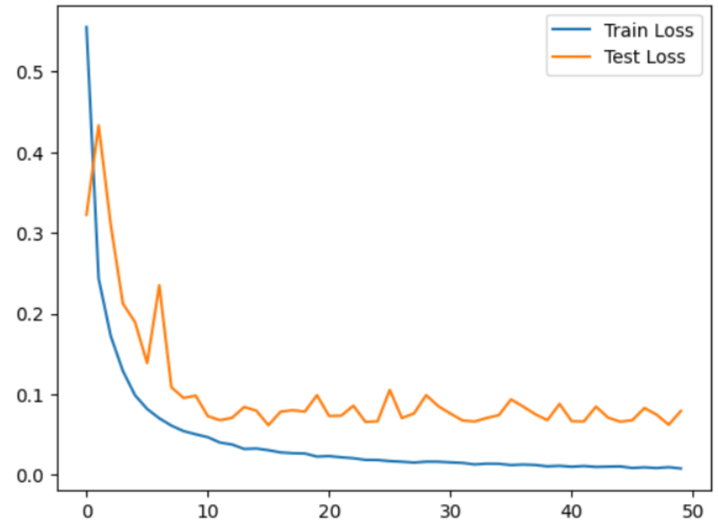
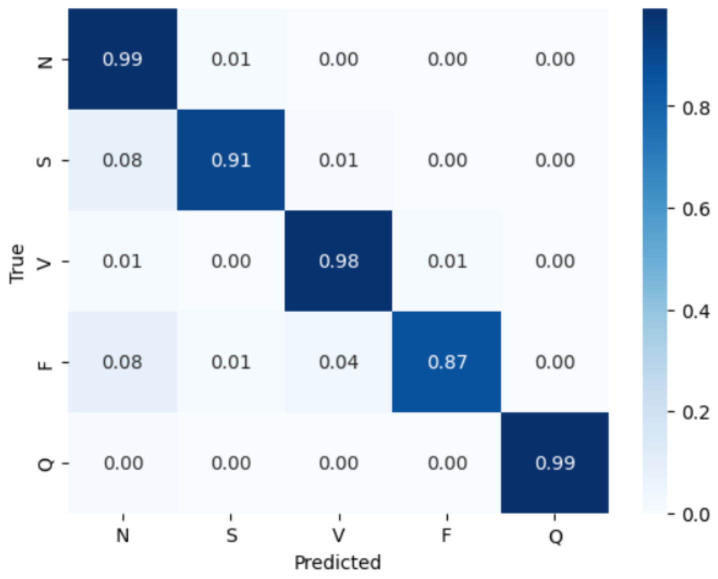


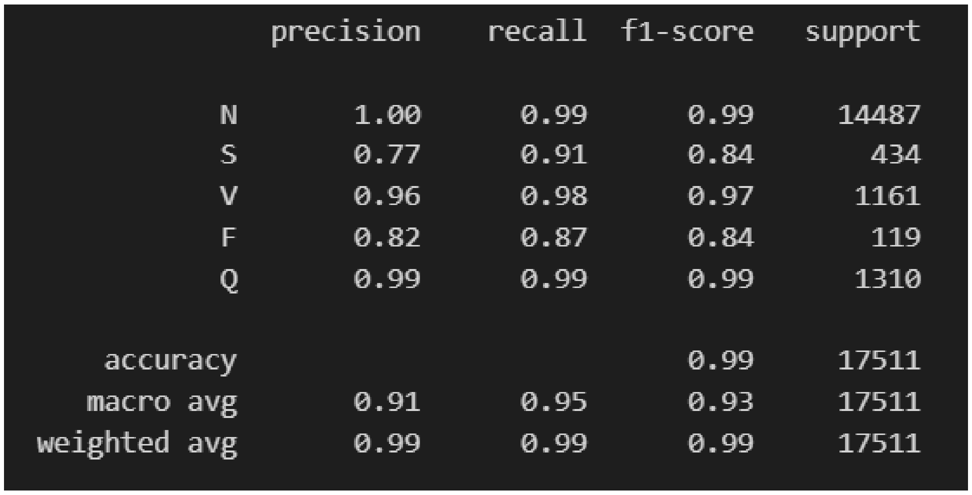
Figure 7. CNN Model Training Loss, Test Loss Graph

In addition, it can be seen from the confusion matrix that most of the samples are classified correctly into their corresponding labels. The model therefore trains excellently on the test set except for very few misclassification cases.



True \ Predicted	N	S	V	F	Q
N	0.99	0.01	0.00	0.00	0.00
S	0.08	0.91	0.01	0.00	0.00
V	0.01	0.00	0.98	0.01	0.00
F	0.08	0.01	0.04	0.87	0.00
Q	0.00	0.00	0.00	0.00	0.99

Figure 8. CNN Model Confusion Matrix



```
              precision    recall  f1-score   support

           N       1.00      0.99      0.99     14487
           S       0.77      0.91      0.84       434
           V       0.96      0.98      0.97      1161
           F       0.82      0.87      0.84       119
           Q       0.99      0.99      0.99      1310

    accuracy                           0.99     17511
   macro avg       0.91      0.95      0.93     17511
weighted avg       0.99      0.99      0.99     17511
```

Figure 9. CNN Model Classification Report

Comparison with Other Existing Models

Comparing our results with that of the models on Papers with code, the best performing model is an ensemble CNN classifier with Domain Adaptation written by Mehmet Yamaç et al. achieving an accuracy of 98.2% . Another paper titled Classification of ECG Arrhythmia using Recurrent Neural Network, by Shraddha Singh et al. used a RNN LSTM for the task achieving an accuracy of 88.1%. The ensemble CNN outperforms the RNN in terms of accuracy, suggesting that it is more suitable for this task. This suggests that there is some spatial dependance in the data that the RNN is unable to capture, along with ensemble classifiers generally being more robust and resistant to overfitting.