Project Reference Number: 46S_BE_1415

Title of the project: An Approach to diagnose Heart Arrythmia disorders using Embedded Machine Learning.

Name of the College & Department: Electronics and Communication Engineering Department, S. G. Balekundri Institute of Technology, Belagavi.

Name of the Guide: PROF. SHANKARGOUD PATIL (shankargoudp@sgbit.edu.in)

Name of the Students:

MR. VALLABH DESHPANDE (vallabhd90@gmail.com)

MR. VINAYAK KELAGINAMANI (vvinayakkelaginamani@gmail.com)

MS. SHWETA NAIK (sshwetanaik05@gmail.com)

MS. SNEHA PATIL (snehapatil0922@gmail.com)

Keywords: Embedded Machine Learning, Heart Arrythmia detection, Edge computing.

Introduction/background:

ECG signal analysis is one of the conventional methods for detecting cardiac arrhythmias. Electrodes are positioned on the patient's chest to detect the electrical activity of the heart and record the ECG readings. The signal is subsequently enhanced, filtered, and recorded for additional examination. Professionals with training analyze ECG signals to search for irregularities in the waveform, such as alterations to the QRS complex or the T wave. These conventional methods, however, have a number of drawbacks, such as the requirement for educated specialists to analyze the signal, the signal's subjectivity in interpretation, and the length of time required for analysis. The current IOT based classification methods rely on the communication and processing of data between devices and servers. However, this can introduce

several challenges and risks that may compromise the accuracy and reliability of the classification results. Some of these challenges are: network latency and bandwidth limitations, server failures and security breaches and Remote location issues. This can affect the quality and availability of the data collected by the devices, which may reduce the accuracy and robustness of the classification models. Hence, this project presents an embedded machine learning approach to classify heart arrythmia in three classes: Normal, Atrial fibrillation and Noise, using Edge Impulse, a platform for developing and deploying machine learning models on embedded devices. The proposed method is optimized for an Arduino Nano 33 BLE Sense development board and the model is trained using a 2017 Challenge training set. It is feasible to develop a machine learning model using the Arduino Nano 33 BLE Sensing board that can precisely identify cardiac arrhythmia in real-time by utilizing the strength of Edge Impulse. It is therefore the perfect foundation for creating lightweight, affordable healthcare solutions that may be applied in distant or resource-constrained environments.

Objectives:

- To create an efficient model to classify various ECG signals with better precision and accuracy.
- To leverage the power of TinyML and deploy the model in low-cost, low-power microcontroller- Arduino Nano 33 BLE Sense.
- To simplify the computation complexity by implementing edge computing standards without internet connectivity.
- To design and implement a 4 neural network architecture that can process electrocardiogram (ECG) signals and output the probability of each class.
- To optimize the model parameters such as learning rate, batch size, number of epochs, regularization and dropout to achieve the best results.

- To provide real-time information with efficient and accurate result.
- Develop a simplified, user-friendly and hand-held interface.
- To validate the model on unseen ECG signals from different sources and conditions and compare the results with existing methods.

Methodology:

In order to create a ML model, we input the .csv files extracted from the datasets, process them and generate useful features. Further, use the NN Classifier and classify the datasets. The flowchart to train the model is described as follows:

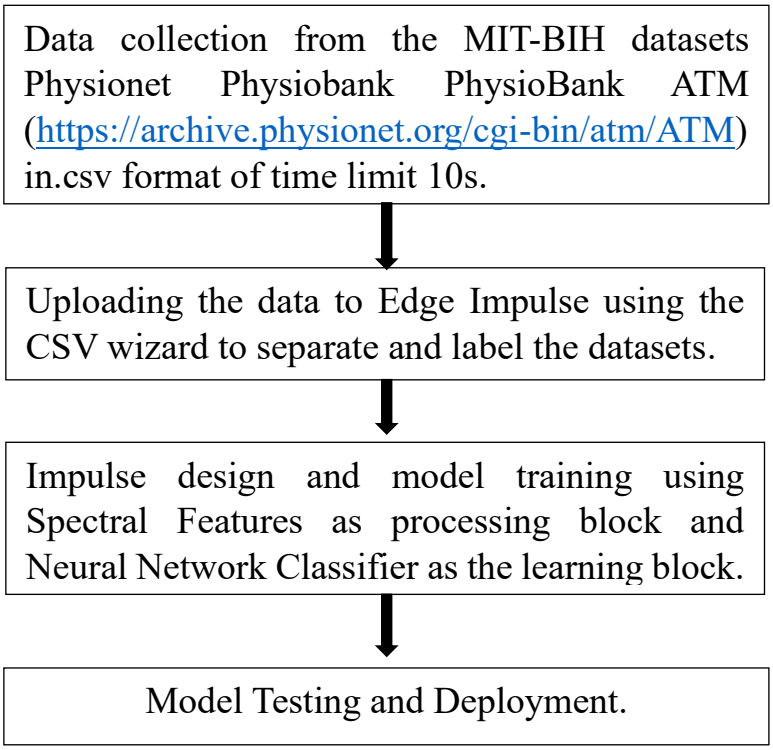


Fig. 1. Model training and deployment flowchart.

A. Data Collection :

The PhysioNet/CinC Challenge 2017 datasets are used from the Physionet Bank ATM. The recordings are split into 3 second intervals from the 10 second individual recordings. In order to explore the data and train the models, the CSV files were then uploaded to Edge Impulse. In order to divide the data into training and testing sets for the model, we used the conventional 80-20 split.

B. Impulse Design :

Three primary building components make up a full Impulse- an input block, a processing block, and a learning block. The following parameters are set onto the block: scale =1, to multiply all raw input values by this number. FFT length is set to 128 and log of spectrum is enables and the FFT frames are allowed to overlap. The extracted features are passed to each layer of your neural network design as inputs, where they are used as building blocks. A SoftMax layer was the final layer employed in the categorization scenario. The likelihood of becoming a member of one of the classes is provided by the last layer.

The Neural network is configured as follows:

- Number of training cycles = 85
- Learning rate=0.0005
- Validation set size=20%

The neural network architecture consists of the input layer with 67 features and 3 successive dense layers with 40, 20 and 20 neurons, followed by a dropout layer of 0.05 respectively.

C. Model Testing and Deployment :

We divided the dataset into training and testing sets before in data collection. The testing set is used to confirm how well the model will perform on data that has not been encountered during training. This will guarantee that the model has not, as is frequently the case, trained to overfit the training data. The Model Training output window provides a summary of the model's results and aids in model evaluation. It is used to assess whether the model can satisfy requirements or whether more hyperparameters and architectures need to be tested.

D. Block diagram :

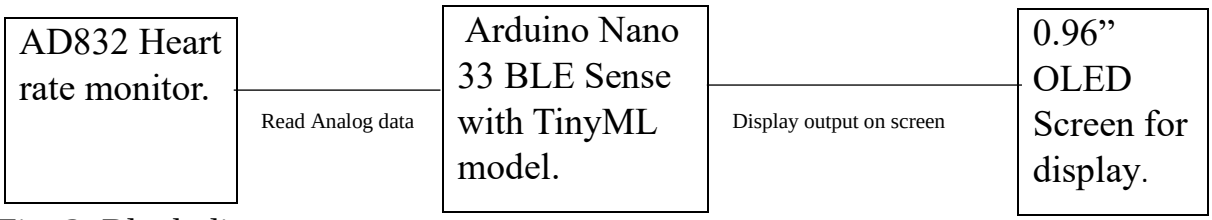


Fig. 2. Block diagram

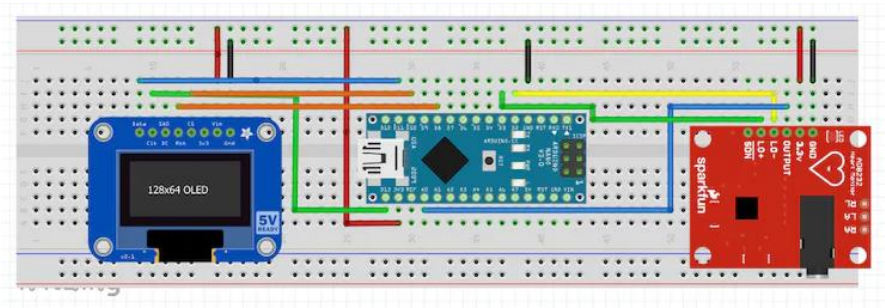


Fig. 3. Connection diagram

The connection for the circuit is as shown above. The OLED screen is connected to the Arduino Nano 33 Ble Sense using I2C protocol, wherein the VCC and GND connections are connected to VCC and GND of the Arduino board respectively, the SCL to A5 and SDA to A4 analog pins. The AD832 Heart Monitor has the ECG signal output at the OUTPUT port, which is read by the A0 pin of Arduino Nano 33 BLE Sense.

Results and Conclusion

The confusion matrix yields an accuracy of nearly 92.6% with a loss of 0.25. The on-board performance is based on the target i.e., Arduino Nano 33 BLE Sense, the output estimations for the inferencing time is 1ms, peak RAM usage is 1.8Kb and flash usage is about 15.4Kbs.

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY
92.6%

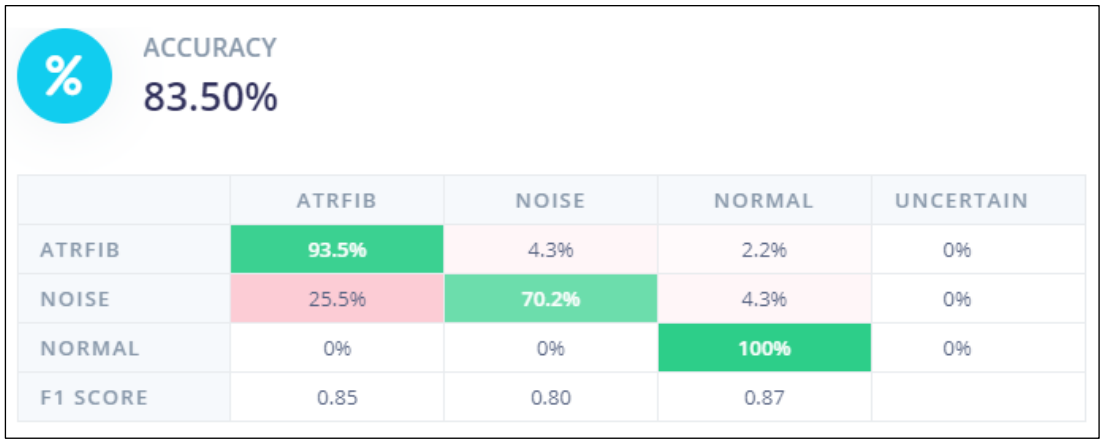
LOSS
0.25

Confusion matrix (validation set)

	ATRFIB	NOISE	NORMAL
ATRFIB	**97.3%**	2.7%	0%
NOISE	7.1%	**89.3%**	3.6%
NORMAL	0%	12.5%	**87.5%**
F1 SCORE	0.96	0.89	0.90

Fig. 3. Model training output

The test datasets contain individual samples for Atrial fibrillation, normal and noise datasets extracted from the datasets. With the limited datasets, the accuracy was observed to be nearly 83.5%.



	ATRFIB	NOISE	NORMAL	UNCERTAIN
ATRFIB	93.5%	4.3%	2.2%	0%
NOISE	25.5%	70.2%	4.3%	0%
NORMAL	0%	0%	100%	0%
F1 SCORE	0.85	0.80	0.87	

Fig. 4. Model Testing output

Live Inferencing Results :

Fig. 5. ECG Analyzer kit

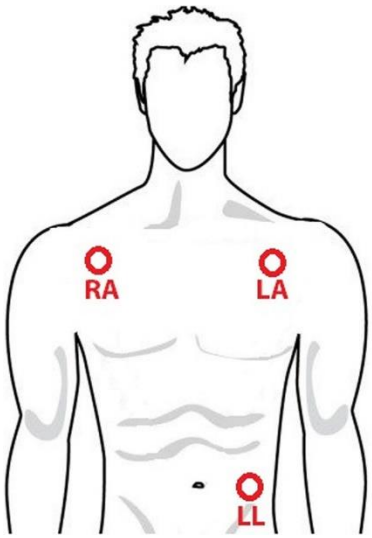


Fig. 6 ECG Analyzer kit attached to subject

The ECG kit is attached to the subject as shown in the adjacent figure. The kit consists of a single lead ECG electrode with three lead – RA (Right Arm), LA (Left Arm) and LL (Left limb). The kit is powered by the USB port attached to the PC/Laptop. The outputs are displayed on the 0.96” OLED Screen. The ECG recordings for a duration of 30seconds are fed into the kit, upon which the ML model performs the classification as Atrial Fibrillation,

Normal and Noise. Thus, the model predicts the live ECG recordings fed to it from the subject as Atrial Fibrillation, Noise and Normal as shown in the below images.

Fig. 7. Prediction of classes : (Starting from left) Atrial Fibrillation, Noise and Normal in %

In this project, we have used embedded hardware and machine learning methods to construct an ECG arrhythmia detection system. To train and evaluate our system, we employed the 2017 PhysioNet/CinC Challenge. Our findings demonstrated that our method has a good accuracy in identifying arrhythmia. Our system was able to operate in real-time without internet access thanks to the utilization of edge computing. Although our approach produced encouraging findings, our study had several drawbacks. The use of a small dataset, which might not reflect the entire population, is one restriction. The usage of a single hardware platform, which could not be transferable to other devices, is another restriction. Our dataset will be expanded, and our system will be tested on many hardware platforms, in further work. We also intend to look at the possibility of using more characteristics and modalities, such photoplethysmography and respiratory signals, to boost the precision of our system.

Innovations in our project:

- Using a low-power microcontroller to run the neural network inference on the device, reducing the latency and bandwidth requirements compared to cloud-based solutions.

- Applying signal processing techniques such as filtering, normalization and segmentation to preprocess the electrocardiogram (ECG) data and extract relevant features for the neural network input.
- Training the neural network on a large and balanced dataset of ECG signals from different sources, using cross-validation and regularization techniques to prevent overfitting and improve generalization.
- Evaluating the performance of the neural network on unseen ECG data, using metrics to measure how well it can classify the 3 classes of heart arrhythmia.
- Implementing a user interface that displays the classification results and provides feedback and guidance to the user, such as alerting them if they have atrial fibrillation or noise in their ECG signal.

Future Scope:

- **Optimizing the machine learning model**: Once the data collection process is complete, the next step is often to optimize the machine learning model by adjusting its architecture, parameters, and hyperparameters.
- **Testing and evaluating the model**: After the machine learning model has been trained and optimized, it is important to test and evaluate its performance on a separate dataset.
- **Integrating the model into an application**: Once the machine learning model has been developed and tested, the next step is often to integrate it into an application or system that can use it to perform a specific task.
- **Deploying the system**: After the machine learning model has been integrated into an application, the next step is often to deploy the system in a real-world setting.
- **Continuing research and development**: There is always room for further research and development in the field of embedded machine learning.