Atrial Fibrillation and Sinus Rhythm detection using TinyML (Embedded Machine Learning)

G.V.B.F Silva[1], M.D. Lima[1], J.A.F. Filho[1] and M.J. Rovai[1]

[1] UNIFEI - Universidade Federal de Itajubá, IESTI, Itajubá - MG, Brazil

Abstract— **Given the various technologies used to measure and detect cardiac arrhythmia, this project proposes using TinyML (Embedded Machine Learning) for atrial fibrillation and normal sinus rhythm classification. The machine learning model is going to be deployed in a microcontroller to bring a small, efficient, and straightforward prototype for the desired purpose. The proposed architecture of the neural model was composed of convolutional networks (CNN), where the input data from the PTB-XL database went through some preprocessing steps, such as filtering and dividing the temporal records into individual heartbeats. The prototype execution in the embedded environment was developed and carried out using the ESP32 development board. The results obtained verified that the model reached an overall accuracy of 94.1% and 94.04% in the training and test stages, respectively. In contrast, it got an overall accuracy of 99.33% in the microcontroller prototype inference, with data extracted from an advanced patient simulator that reproduces different cardiac signals.**

Keywords— **TinyML, Machine Learning, R-wave detection.**

I. Introduction

Atrial fibrillation (AF or AFIB) is the most common form of clinically crucial cardiac arrhythmia [1]. Its predominance increases with age and is often associated with structural cardiac diseases [2].

Remote monitoring has excellent sensitivity (95%) in detecting AF. Furthermore, the potential benefits of remote monitoring include early detection and reaction to prevent serious adverse events related to AFIB [3].

Therefore, a wide variety of technologies serve as the basis for the operation of various remote monitoring systems, which usually involve concepts of IoT (Internet of Things). With that, a newly emerging technique known as TinyML, makes possible the use of microcontrollers with limited memory and little processing power for automated tasks of classification with machine learning.

The great advantage of using a system with TinyML for cardiac monitoring is the low power energy needed for its operation. At the same time, it remains operating with a high degree of accuracy and classification speed. Furthermore, this solution can offer other benefits such as low latency, reduced bandwidth, reliability, and greater security [4], which are exciting features for the purpose addressed.

II. Proposal

The goal of the research is to develop a prototype using TinyML, capable of detecting whether a patient has AF or has a healthy heart rhythm, known as normal sinus rhythm (SR), and deploy the model in a microcontroller. The study was carried out with a dataset publicly available on the Phisionet Platform and also with the SIM-Man patient simulator. In this work, no experiments were performed with humans or animals.

The detection needs to be done using an electrocardiogram (ECG[5]) measurement with just one lead [6]. Therefore, the project will involve everything from the data preparation stage to train the machine learning model to the measurement of heart rate, treatment of sampled data, and classification of the model. This model, in turn, should be small enough to run on a 32-bit microcontroller, which has a reduced amount of memory, which in this project will be the ESP32 [7], but it should also have a good efficiency to classify the heart rates.

The prototype developed for carrying out the measurements related to the model inference is composed of the following main components: ESP32, AD8232, ECG electrodes and cable, ADS1115 ADC, and a 4 terminal touch switch.

The prototype works from the measurement of cardiac signals through the AD8232 and the conversion of the analog signal to digital with the ADS1115. The model execution and digital data processing are performed through ESP32. Figure 1 shows how these components are connected.

III. Development

A. Electrodes Positioning

The connection of the electrode pads with the signals to be sampled from the body, according to the type I lead, is done

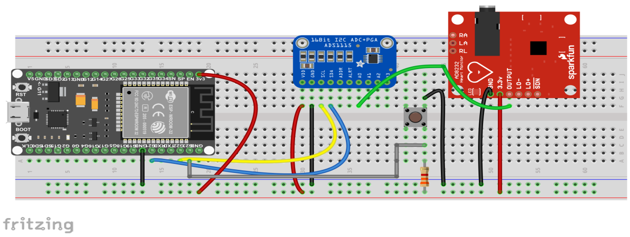

Fig. 1: Prototype assembly schematic, via Fritzing, 2021.

as follows: Green cable – LA (Left Arm); Red cable – RA (Right Arm); Yellow cable – RL (Right Leg).

B. *Database and data processing*

The block diagram representing the development of the project as a whole is shown in the Figure 2.

Thus, the first phase of the project was to identify a dataset that contained the SR and AFIB classes.

Among the numerous databases available, two stand out in particular, where both consist of a large group of ECG records. Are they: a) The ICBEB2018 dataset, containing 12-lead 10s ECG records of 10,646 patients with a sampling rate of 500 Hz that presents 11 heart rhythms and 56 types of additional cardiovascular conditions [8]. b) The other alternative found was the PTB-XL database, whose dataset comprises 21,837 clinical records of 12-lead ECG with a duration of 10 seconds from 18,885 patients. Data is available at frequency rates of 100 and 500Hz [9].

The main strengths of the PTB-XL database are that the records reflect different levels of ECG data quality in real-world measurements, thus serving to properly assess the performance of machine learning algorithms towards efficiency against changing conditions against various imperfections in the input data.

Thus, according to the advantages presented above, it was chosen to use the PTB-XL database as the reference data provider for this project, more specifically, the database with a sampling frequency of 100Hz, for the reason of implying less data-heavy and less memory usage, which is critical in a TinyML project.

The next step consisted of loading the database in Python and separating only the data that you would like to work with, that is, those related to SR and AFIB heart rates. Concerning data separation, the first procedure performed was to select only the data from Lead I, as it was sufficient for the detection of AF and SR, in addition to being a more economical solution compared to a 12-lead ECG [6].

Now, it was important to match the number of records of each heart rhythm that was initially unbalanced, with 16,782 SR and 1,514 AF. This is done to remove any type of bias that could interfere with the way the model performs its classifications so that there is an equal division (balancing) between the two classes (Downsampling).

The codes that present step-by-step the activities carried out in the project are presented in the GitHub repository of our project [10].

With the database prepared, it was now possible to start processing the data, capture its main characteristics and send them to the model. It was known that initially, the data correspond to a time series of 10s each, however, this direct format is difficult for the model to work with.

According to the literature on machine learning models applied to cardiac monitoring in general, there is a tendency to split the complete raw ECG signal, which consists of time series data, into a group of individual heartbeats. With this technique, results were obtained that even surpass other ECG classifiers that follow more complex resource selection approachess [11].

To separate the raw signal into a set of single heartbeats, it was necessary to carry out some pre-processing steps, starting with filtering the ECG records, to eliminate noise that could make it difficult for the model to identify heart rhythms.

In this way, the filtering of the ECG signal was made through the application of 2 filters: a high-pass filter and a low-pass filter. For the high-pass filter, a 4th-order Butterworth filter and a 0.5Hz cutoff frequency were used. Similarly, a low-pass Butterworth filter of order 3 and cutoff frequency of 41Hz was used. The choice of cutoff frequencies was made to model the values applied by the AD8232 filter, which performs a previous filtering in the data sampling [12].

A full explanation of the signal filtering process can be seen in the GitHub repository of our project [10].

Once filtering has been done, the next step was to identify the R peaks at the individual heartbeats, where the R peak corresponds to the largest QRS complex value of a heartbeats.

As in this project, the algorithm will be applied not only on computer (training) but also on the microcontroller (inference), it was clear that the algorithm responsible for finding the R peaks needs to be light and fast enough not to present compatibility problems in the embedded environment. Knowing this, it was proposed to create a program that would be able to identify the R peaks of an ECG record quickly and efficiently, so that it was not heavy.

The algorithm created works by sweeping the entire signal, looking for the highest value point within a window defined by a reference value. This reference value is equal to half of the largest absolute value of the signal. Details about the algorithm are presented in the GitHub repository [10].

With the values of the R peaks and their positions, it was

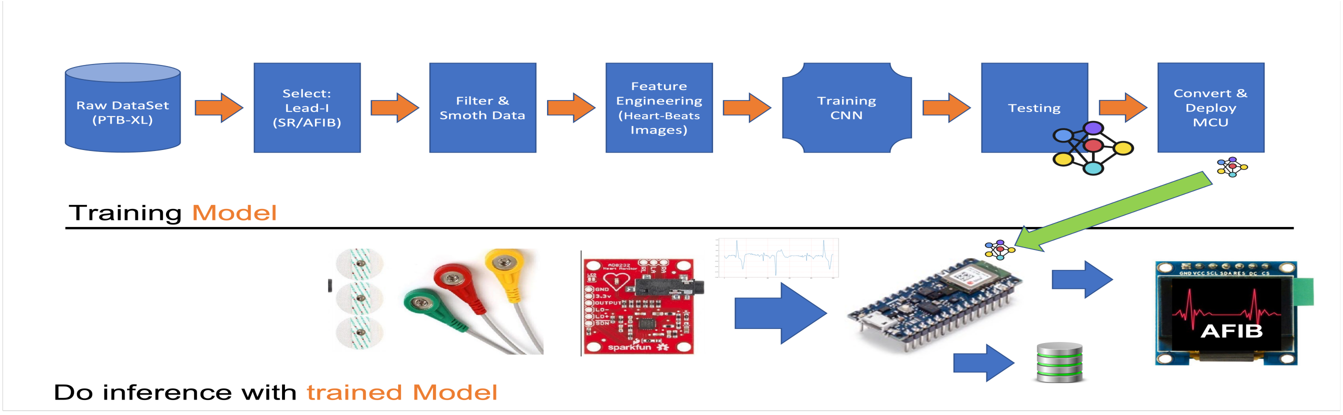

Fig. 2: Project block diagram., via Marcelo Rovai, 2021.

possible to obtain the individual heartbeats that have 100 points, according to the sampling frequency of 100Hz. It was enough to take the position of each R peak found and separate 100 samples around it, 40 before and 60 after. This way, it will be guaranteed that you will have a complete heartbeats made up of all the waves that make up one. One examples of a set of signals of an individual SR heartbeats obtained from the created function are shown on Figure 3.

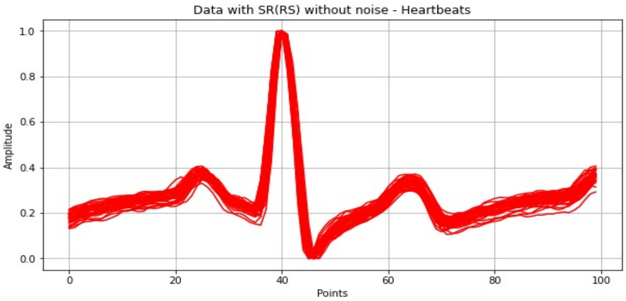

Fig. 3: 39 superimposed SR single heartbeats signals, via Author, 2021.

With the individual heartbeats already obtained, the last stage of data pre-processing was made, which is normalization, to avoid the problem in which the model to be trained may have a bias towards distorted or anomalous values. For the heartbeats to have values at the same level, it was preferred to apply individual normalization.

C. Machine learning model

As the project works with individual heartbeats (time series data), the chosen model will be based on the convolutional neural network (CNN). This model was chosen mainly because it was an architecture with proven effectiveness in medical applications involving ECG classifications [13] [14].

The first phase of the training process was performed in the Jupyter Notebook tool [15], where the metrics used to define the quality of the focused model for TinyML were: accuracy, memory consumption, and latency.

Accuracy was determined by the confusion matrix obtained after training, which generally showed the model's performance in correcting its predictions, while memory consumption was directly affected by the model's size, which in turn was influenced by the number of parameters that make up the own.

Thus, it was necessary to find a balance between the model's performance and its size, to adapt to the memory and processing specifications of the microcontroller used.

Taking these considerations into account, the model that managed to achieve the best performance is presented on Figure 4.

The model is initially composed of 1D CNN layers, which are more suitable than 2D CNN layers for real-time applications on low-power and low-memory devices [13]. In addition, the 1D Max Pooling technique was applied in order to make the data leaner. In addition to these, there was also the application of Dropout, with the aim of reducing possible Overfitting. Then was applied a Flatten layer to transition from the convolution layers to the Dense ones. These, in turn, will completely connect the neural networks, followed by a Softmax activation function for final classification.

Since the most efficient model had already been identified in Jupyter Notebook, the same model was implemented on the Edge Impulse platform [16], to first know the latency value that was available but also to check whether it was possible to achieve further optimization concerning memory consumption and efficiency, which has happened.

After that, a test of the model was carried out with data from the PTB-XL which were different from the one em-

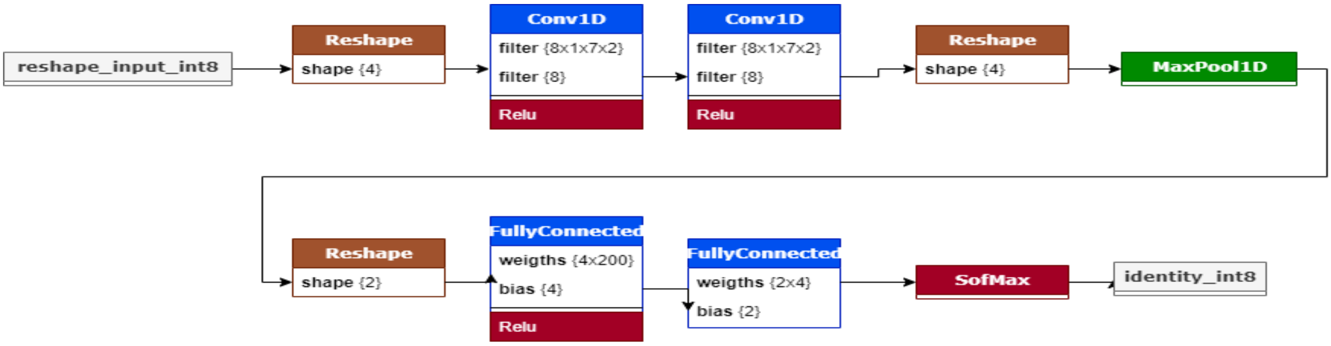

Fig. 4: Neural model, via Author, 2021.

ployed in training, to verify if it maintained the training efficiency even with completely new data, which was also proven. Similarly, with the same objective of proving the efficiency and functionality of the model, data that were not part of the original database were collected and applied to the model to make the predictions. These data were particularly obtained from a patient simulator [17] that is capable of reproducing different heart rhythm conditions, including SR and AF.

D. *Model Conversion and Inference*

After the test step, it was then possible to optimize the model and convert it from Python to C++.

Devices for TinyML applications do not have a high processing capacity. It was necessary to perform optimizations in the trained model such as making the storage size smaller, through quantization [18], in which 32-bit float numeric parameters are converted to 8-bit integers. This practice is possible through the use of the TensorFlow Lite framework [19].

In this way, in the model converted to C++, a significant optimization of memory usage is achieved, which will be useful in the implementation in the microcontroller.

For the development of the embedded code, the method used was to reapply all the steps carried out in the training of the model written in Python. For this purpose, the development of the code in C++ was divided into the following steps: acquisition and processing of data in real-time, separation of beats, the introduction of the signal in the model, and classification through the model provided by the Edge Impulse library. For more in-depth details on how embedded code works, see the GitHub repository of our project [10].

E. *Prototype evaluation methodology*

After a model is trained, its performance is measured through precision and accuracy metrics, more specifically, accuracy, recall, and F1-score [20] and a confusion matrix, which describes the performance of the classifications performed by the model [21]. As we are working with binary classification, the confusion matrix of this project will have a dimension of 2x2.

In training, 80%(27526 single heartbeats) of the total dataset is used, where this set is used to assign the weights and biases that go into the model.

In the test group, 20%(6882 single heartbeats) of the data is used for the final assessment. Having never seen this dataset, the model is free from any bias.

The division of training and test data was performed by a Python function (train-test-split) that distributes the data from the initial set at random in the proportion chosen, in order to avoid a bias.

Finally, the performance measurement of the model already converted to C++ running on the ESP32 microcontroller was as follows: a) the measurements were made from the cardiac signals of SR and AF of the patient simulator [17], in a total time of 50 minutes for each heart rate and, b) 120 measurements of 10s each were taken with heart rates ranging from 40 to 110 bpm in the case of SR measurements and from 40 to 170 bpm in the AF measurements.

The model then made the classifications of heartbeats obtained from the 10s and with these values, it was analyzed which rhythm had the highest incidence of predictions. In other words, the average duration of each exam was 25s, meaning that the duration of data pre-processing was approximately 15s for each exam.

IV. Results

Regarding the training of the model on the Edge Impulse platform, in which there was a total of data equally divided between SR and AFIB which together corresponded to 38 minutes and 10 seconds, Table 1 shows the result.

Table 1: Training Data Confusion Matrix

	AFIB	SR
AFIB	95,8%	4,2%
SR	7,7%	92,3%
F1 SCORE	94%	94%

That is, it appears that the model was able to correctly predict (true positive) that the patient had AF (AFIB) in 95.8% of the number of times it predicted this condition, while it was correct that the patient had SR (SR) in 92.3% (true negative) in the same situation. The result of the calculations for the recall was: recall (AFIB) = 92.56% and recall (SR) = 95.65%.

About the F1-score, it can be seen in the table previously available that both the AFIB and SR classes presented a result of 94%, in addition to an overall accuracy of 94.1%, which measures all input data, which the model managed to predict correctly.

It was also possible to check the model's performance parameters, which are the amount of RAM and FLASH memory used and the latency (Table 2).

Table 2: Model parameters

INFERENCING TIME	PEAK RAM USED	FLASH USED
10 ms	5,3K	69,4K

In other words, the quantized model is expected to take around 10ms of inference time, using 5.3KB in RAM and 69.4KB in ROM (FLASH). Concerning latency, it was clear that the model can make the inference almost instantaneously. As for memory consumption, it was noted that they are below the limits specified by the ESP32 microcontroller, which has a RAM memory of 520KB and a FLASH memory of 4MB. In other words, the model can run smoothly on the board, since it respects its memory limits.

Table 3 shows the model results from the test data.

Table 3: Test Data Confusion Matrix

	AFIB	SR
AFIB	95%	5%
SR	7%	93%
F1 SCORE	94%	94%

Analyzing the confusion matrix, it appears that the training result was maintained for the test since the general accuracy remains at 94.04% and the accuracy of each class is preserved above 90%.

Table 4 shows the results of the prototype running on the ESP32 microcontroller.

Table 4: Real Data Confusion Matrix

	AFIB	SR
AFIB	100%	0%
SR	3.33%	96.67%
F1 SCORE	98.36%	98.31%

According to the results obtained through the confusion matrix, the following parameters were calculated for SR and AFIB: Recall (AFIB) = 96.78%, Recall (SR) = 100% and Overall accuracy of 99.33%.

These results show the quality of the model in identifying the treated heart rhythms even with completely new data for the model.

V. Conclusion

Regarding the results, there was a great performance of a model built to not consume too much memory, reaching an overall accuracy above 90% both in training and testing as well as in measurements performed with ECG signals coming from a patient simulator. These numbers are as high as the results obtained using state-of-the-art convolutional models, such as [11], which achieved an overall accuracy of 93.6%, and [22] which achieved an accuracy of 92.7%. The detail is that these models are more complex and heavy than the one developed in this work, meaning that they would probably not run on microcontrollers.

Therefore, the main point confirmed by the work that differentiates it from the other analyzed articles was the use of the TinyML technique that allows the classification algorithm to be embedded in a small processor. Despite being a new technology, it already shows its potential mainly in the medical area, in applications where processing and memory are critical.

In addition, it was observed that the constructed model obtained superior results in relation to other articles [14] [23] that used the CNN 1-D architecture as the basis of the model and obtained an accuracy in the AF classification of respectively 80.8% and 82%. The main difference between the studies was that ours employed additional pre-processing steps, such as filtering and splitting the raw signal into R-wave-centered heartbeats, which made the model input data more suitable for classification. This practice proved important when dealing with simpler models, where the quality of the input data is crucial for good performance.

With the execution of this work, it was concluded that obtaining an efficient, small and simple machine learning prototype that works successfully in a microcontroller such as ESP32 is not only feasible but also viable, even for a critical issue that involves human health. It was possible to analyze, through an end-to-end project, all the main steps that involve machine learning work in embedded devices, from the definition of the objective, passing through the definition of the database and its treatment, building and training the neural model, converting and optimizing this model to run it on a microcontroller, and finally testing it with real-time measurements of external data so that it was possible to recognize the importance of each in the general context since all the steps are closely linked to one another within a cyclical process.

The caveat that can be cited is about the performance of the model with SR heart rate signals with very high heart rates, where a certain drop in the overall performance of the converted model was noted.

Finally, the project can assume certain improvements such as increasing the number of arrhythmias to be classified, in addition to improving the quality of the database to cover a greater number of possible situations of data samples.

The dataset PTB-XL and the patient simulator (Figure 5) were fundamental for the development of the model. With the results obtained, it is now possible to proceed with great confidence to a clinical trial in human beings.

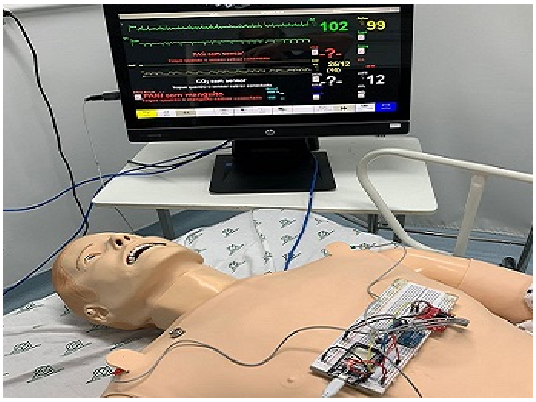

Fig. 5: Prototype Inference, via Author, 2021.

References

1. Justo Fernanda Augusto, Silva Ana Flávia Garcia. Aspectos epidemiológicos da fibrilação atrial *Revista de Medicina.* 2014;93:1–13.
2. Manenti Maitê Thomazi, others . DESENVOLVIMENTO DE UM PROTÓTIPO DE MONITORAMENTO DO SINAL ELÉTRICO CARDÍACO E DIAGNÓSTICO DE FIBRILAÇÃO ATRIAL -. 2018.
3. Lopes Marcelo Antônio Cartaxo Queiroga, Oliveira Gláucia Maria Moraes de, Ribeiro Antonio Luiz Pinho, et al. Diretriz da Sociedade Brasileira de Cardiologia sobre Telemedicina na Cardiologia–2019 *Arquivos Brasileiros de Cardiologia.* 2019;113:1006–1056.
4. Warden Pete, Situnayake Daniel. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media 2019.
5. Bhanu HS, Tejaswini S, Sahana MS, Bhargavi K, Praveena KS, Jayanna SS. Analysis of ECG Signal and Classification of Arrhythmia in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*:619–623IEEE 2021.
6. Duarte Rui, Stainthorpe Angela, Mahon James, et al. Lead-I ECG for detecting atrial fibrillation in patients attending primary care with an irregular pulse using single-time point testing: A systematic review and economic evaluation *PloS one.* 2019;14:e0226671.
7. Babiuch Marek, Foltỳnek Petr, Smutnỳ Pavel. Using the ESP32 microcontroller for data processing in *2019 20th International Carpathian Control Conference (ICCC)*:1–6IEEE 2019.
8. Zheng Jianwei, Zhang Jianming, Danioko Sidy, Yao Hai, Guo Hangyuan, Rakovski Cyril. A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients *Scientific data.* 2020;7:1–8.
9. Wagner Patrick, Strodthoff Nils, Bousseljot Ralf-Dieter, et al. PTB-XL, a large publicly available electrocardiography dataset *Scientific Data.* 2020;7:1–15.
10. GitHub repository https://github.com/Gui7621/TFG-AFIB-and-R-detection-using-ML-in-embedded-systems 2020.
11. Alfaras Miquel, Soriano Miguel C, Ortín Silvia. A fast machine learning model for ECG-based heartbeat classification and arrhythmia detection *Frontiers in Physics.* 2019;7:103.
12. Inc Analog Devices. *DATASHEET AD8232* 2021.
13. Kiranyaz Serkan, Avci Onur, Abdeljaber Osama, Ince Turker, Gabbouj Moncef, Inman Daniel J.. 1D convolutional neural networks and applications: A survey *Mechanical Systems and Signal Processing.* 2021;151:107398.
14. Hsieh Chaur-Heh, Li Yan-Shuo, Hwang Bor-Jiunn, Hsiao Ching-Hua. Detection of atrial fibrillation using 1D convolutional neural network *Sensors.* 2020;20:2136.
15. Jupyter Notebook https://jupyter.org/index.html 2021.
16. Zach Jan. Edge Impulse https://www.edgeimpulse.com/ 2021.
17. LAERDAL . SimMan https://laerdal.com/br/products/simulation-training/emergency-care-trauma/simman-3g/ 2021.
18. TensorFlow . Post-training quantization https://tensorflow.google.cn/lite/performance/post_training_quantization 2021.
19. TensorFlow . TensorFlow Lite converter https://tensorflow.google.cn/lite/convert/?hl=en 2021.
20. Jason . How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/ 2020.
21. Visa Sofia, Ramsay Brian, Ralescu Anca L, Van Der Knaap Esther. Confusion matrix-based feature selection. *MAICS.* 2011;710:120–127.
22. Rajkumar A, Ganesan M, Lavanya R. Arrhythmia classification on ECG using Deep Learning in *2019 5th international conference on advanced computing & communication systems (ICACCS)*:365–369IEEE 2019.
23. Xiong Zhaohan, Stiles Martin K, Zhao Jichao. Robust ECG signal classification for detection of atrial fibrillation using a novel neural network in *2017 Computing in Cardiology (CinC)*:1–4IEEE 2017.

Author: Guilherme Vilas Boas Ferreira da Silva
Institute: UNIFEI - Universidade Federal de Itajubá, IESTI
Street: Avenida BPS, 1303 - Pinheirinho
City: Itajubá
Country: Brazil
Email: guifdasilva@yahoo.com.br