Multi-class classification from single lead ECG recordings

gitBrexit♠ gitMelk♣

***Abstract*—The automatic classification of heart rhythms using short time single lead ECG recordings is a challenging task that has been widely studied recently.**
In this paper we present our work that aims at classifying these kind of ECG signals as Atrial Fibrillation (Afib), Normal, Other rhythms or too noisy to be classified (Noisy). We developed three different binary classifiers as Recurrent Neural Networks (RNNs) both with a binary cross-entropy loss function and a weighted version of it. We used these three RNNs to develop a cascade classifier for the samples of the given dataset, considering the problem as a multiple binary classification problem.
We obtained similar results, with a slightly better result using the unweighted loss function, with an accuracy of 81.18% vs 80.01% and a F1 score of 0.77 vs 0.76.

***Index Terms*—ECG, arrhythmia, RNN, multiclass, single-lead, hierarchical model**

I. Introduction ♠

Atrial fibrillation (Afib) is a heart condition in which the electrical activity of the atria is rapid and irregular; being one of the most common cardiac rhythm disturbances, it represents one of the major causes for cardiac morbidity and mortality [1]. The risk of Afib presence is positively correlated with the increasing age of the patients: as the population of elderly individuals tends to grow, also does the number of patients suffering from this type of arrhythmia and the effort on trying to prevent the correlated risks (i.e. stroke prevention and rhythm management) [2].
The recent progresses in technology allowed the development of mobile tools able to detect and record heart activity. Both the computational power and the enhancement in connectivity are essential for the diffusion of low-cost and widely used devices, which also need to be as accurate as possible.
At the moment, the classification of ECG rhythm is generally done by cardiologists, who manually review the recordings and search for abnormalities: this is expensive both from a time and money point of view, and it may not be as accurate as we would it to be. The development of mobile devices, which can also record single lead ECG signals, pointed out the need for automated detection and classification of heart conditions and rhythms. In this way the possibilities of prevention and real-time monitoring would have better outcomes, not to mention the savings that would result by the automation of the procedure [3].
The challenges in the diagnosis of Afib are mostly represented by the presence of noise in the recordings, especially when the signals are acquired using techniques such as the fingertips acquisition, and by the limited availability of samples, which are often lasting only few seconds [4].
Therefore, the aim of our study is to present a simple and effective way to tackle these challenges: we classified hearth rhythms, based on the dataset provided by *PhysioNet* for the *2017 Computing in Cardiology Challenge* [5], using a Recurrent Neural Network (RNN). We developed a technique for the classification of Afib, Normal, Other or Noisy rhythms, following the idea of a *One-vs-All* strategy.
The rest of the paper is structured as follows: in the next section we present the previous works related to the challenge and the ones that inspired us during our project; in Section III we give a general view of the whole idea behind our project; in Section IV we describe in details the original provided data, our processing pipeline and how we built our datasets; in Section V we present our RNN model, the loss functions used and the final cascade classifier; finally, the results and the concluding remarks are shown in Sections VI and VII. We also added at the end a special subsection with the description of the difficulties and other problems we encountered during our work.

II. Related Work ♣

To get an idea of the project we had to develop, we firstly read the *PhysioNet 2017 Challenge* final paper [5] to understand what kind of data we were working with, the scoring to check the quality of a project and the end results of the researchers competing with their teams.
Then we read the paper of Warrick *et al.* in [6] which had one of the best results using a CNN with some Convolutional and Long-Short Term memory (LSTM) layers. From their work we also acknowledged the class imbalance problem: the samples classes have a significantly disparity in the dataset.
Very good results were obtained using the spectrogram of the signals, in [7] Zihlmann *et al.* made use of the logarithmic spectrogram of the ECG recordings. In their pre-processing, they did some data augmentation before feeding their models consisting of a CNN and a CRNN. The two models have a different number of Convolutional layers, but the main difference is that the CRNN also use a LSTM layer and it has a final F1 score higher than the first one.
Another project based on spectrograms is [8] by Rubin *et al.*, their final F1 results were high too and there was the peculiarity of a post-processing we didn't find in other projects.

After they trained their CNN made of more than 40 layers, they built an algorithm to classify the rhythm as Normal or Other in case the discrimination of the CNN model can not be determinant between the two classes.
The project we started before we developed the final one was based on the spectrograms of the ECG signals. To deal with the class imbalance problem we implemented what Park *et al.* did in a recent paper [9] for automatic speech recognition, obtaining state-of-the-art results in that field. Even if it was originally meant for audio signal, we wrote our own code to add more data using ”time warping” and ”frequency masking” following their methods. Unfortunately, this project did not provide significant results, so we abandoned it.
Our final work was inspired by the one presented for the *PhysioNet challenge* by M. Limam and F. Precioso in [10], where they used the signals in the time domain. In this project, they implemented a hierarchical model consisting of three CRNN fed with the time-signals: firstly it classifies Afib versus the rest of the classes, then, after it discards all the Afib samples, it classifies the Normal class against the others. The third time it classifies the Other and Noisy classes after discarding the Normal class. We implemented our own multiple binary classification problem based on their work, but their CRNN was not ideal in our case and we decided to use another model which is what we present in the following pages.

III. Processing Pipeline ♠♣

In this section we briefly describe our processing pipeline, just to give a general idea of our concept.
First we extracted and loaded the raw ECG data from the given compressed file along with the correspondent labels.
Our main idea for the classification is to build a hierarchical model made by three binary classifiers trained with three different datasets. They work as a simple cascade classifier, each one estimating one out of the four possible labels.
In order to do so, we first split the data in *Training* and *Test* set and then we created three different datasets only using the *Training* set as follows:

- ’Afib’ class (labelled as 1) against all the others (labelled as 0)
- ’Normal’ class (labelled as 1) against ’Other’ and ’Noisy’ classes (labelled as 0)
- ’Other’ class (labelled as 1) against ’Noisy’ class (labelled as 0)

And then, again, we divided each of these datasets in *Training* and *Validation*, in order to use them during the training and validation of the RNN models.
In Fig. 1 we present a visual representation of the datasets division explained above.

Next, we defined multiple functions for the preprocessing phase which are applied to the already mentioned datasets. A detailed explanation for each of them will be given in the next paragraphs.
After defining the RNN architecture, we trained it three times: each model we obtained was trained with one of the datasets.

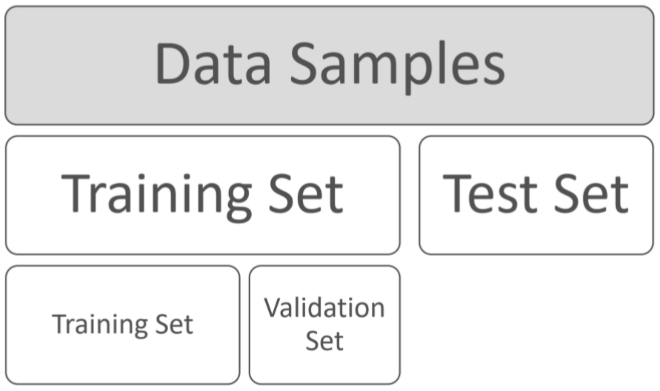

Fig. 1: Visual representation of the dataset division.

In this way the first model was used to find the *Afib* samples, the second the *Normal* ones and the final one classified the remaining samples as *Other* or *Noisy*.
Finally, we defined a function that implemented the cascade classification: given the three trained model, it returned the final predicted labels from the *Test* set.
To check the quality of our algorithm, we also computed some basic performance metrics and we plotted them.
In Fig. 2, for a clearer explanation, we provide a diagram containing the various blocks of our pipeline in their correspondent order.

IV. Signals ♠♣

A. Provided dataset and labels

The dataset was provided by *PhysioNet* during the *2017 challenge* [5]. Each of the 8528 samples was recorded using an ”AliveCor” single channel ECG device, which records the user’s ECG signal using the fingertips. The data were stored at 300 Hz and quantized at 16-bit, they have a bandwidth of 0.5-40 Hz and a dynamic range of $\pm$ 5 mV. The recordings last from 9s to 61s.
The labelling was firstly done by a pool of experts, and then refined three times. We used the third and final version of the labels which is supposed to be the most accurate.

B. Preprocessing

For the processing of the raw ECG signals, we adopted some auxiliary functions. Beside the methods needed for correctly loading the data, we used the following three algorithms:

- **Baseline wander removal**: as the ECG signal could contain artefacts such as breathing, electrical disturbances caused by charged electrodes or the patients movements, we removed the low-frequency trends as they are related to these factors. Since the ECG signal also has some low frequency components, for their preservation we adopted two median filters with a window size of 60 and 180 samples respectively. These were used to estimate the baseline, which was then removed from the signal: the resulting signal is almost centred to zero.

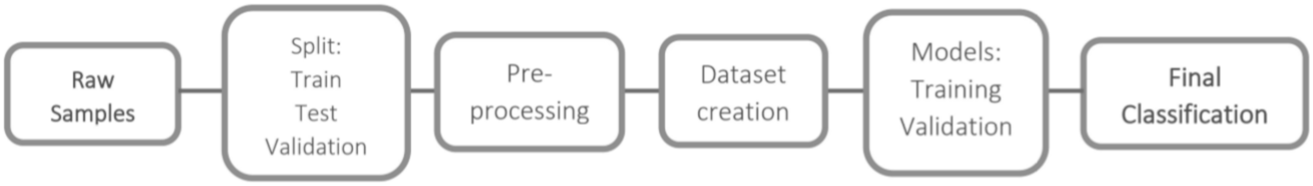

Fig. 2: Final classifiers hierarchy

- **Normalization**: this function has been utilized because the amplitude of the ECG is affected by many subject related factors, e.g. skin resistivity, or by the quality of the electrodes connections. Since we are more interested in the shape of the signal rather than its amplitude, we applied the normalization independently on each sample. For this purpose we estimated the amplitude of the signal at the 99^{th} and 5^{th} percentile, to exclude outliers: their subtraction represents the normalization factor. The signal, divided by the normalization factor is the final output: its range in amplitude is almost between -1 and +1.
- **Random crop**: another problem we encountered while developing the NN is the difference in length of the signals. As our NN works only with signals of the same size, we fixed their length to 9000 samples. The oversized signals were thus cropped, while the shorter ones were zero-padded. In particular, for the crop in the *Training* set we randomly selected the 9000 samples window. In the *Validation* and *Test* set, instead, we centred the window in the middle of the signal in order to perform the validation/test always on the same data, without any randomness.

C. Datasets

As already mentioned in section III we built a *Training* and *Validation* dataset for each trained NN. The latter was made by the 20% of the samples, the remaining 80% was used as *Training* set. In TABLE 1 we describe their numerical composition.

Dataset	Label	Train	Validation
Afib	A (1)	485 (8.89%)	121 (8.86%)
	N+O+∼ (0)	4972 (91.11%)	1244 (91.14%)
Normal	N (1)	3248 (65.33%)	813 (65.35%)
	O+∼ (0)	1724 (34.67%)	431 (34.65%)
Other	O (1)	1546 (89.68%)	386 (89.56%)
	∼ (0)	178 (10.32%)	45 (10.44%)

TABLE 1: Train and Validation data division. The A label corresponds to Afib, N to Normal, O to Other and ∼ to Noisy. For each of the two classes (0 and 1) both the number of samples and its percentage are reported.

For the final labels estimation we used the *Test* dataset made from the 20% of the original whole dataset. The *Test* set is composed as follows:

- **Normal ECG**: 1015 (59.50%)
- **Afib ECG**: 152 (8.91%)
- **Other ECG**: 483 (28.31%)
- **Noisy ECG**: 56 (3.28%)

During the datasets creations, the samples were sampled in a different order in every training epoch for the *Training* set, but not for the *Validation* and *Test* set to assure randomness avoidance and be able to compare the results.

V. Learning Framework

A. Model ♠♣

The pre-processed data was fed to our RNN through an input layer with input shape (9000,1), corresponding to the size of the cropped ECG signal. Then we adopted a series of three one-dimensional Convolutional layers, all of them with kernel size equal to 9 and 32, 64, 128 channels respectively. Every Convolutional layer is followed by a Batch Normalization layer and an Activation layer. The former normalizes the activations of the previous layer at each batch and it is used to keep the mean activation close to zero; the latter simply applies an activation function to the output, in our case a "ReLu" activation function.
This series of Convolutional layers is needed in order to reduce the dimensionality of the input: for this purpose the strides dimension was set on 4 and this also allowed us to feed the subsequent Gated Recurrent Unit (GRU) layer in an efficient way. The GRU layer has 128 units and it is followed by a Flatten layer which simply flattens the input and then a Dense layer which applies a linear operation on the layer's input vector.
We described the used RNN and its parameters in TABLE 2.

B. Loss function ♠

To try to cope with the unbalanced datasets (as we can see in TABLE 1), we developed and compared the results of two types of loss functions.
We used both a *binary crossentropy* loss function and a weighted version of it. We computed the 'balanced' class weights as:

$$\frac{N_s}{N_c \cdot N_{occ}} \tag{1}$$

where N_s is the total number of samples in the dataset, N_c is the number of different classes and N_{occ} the number of occurrences of the samples for the given class.
In this way we obtained the following weights for the three datasets, for the 1 and 0 class respectively:

Layer	Parameter
Conv1D	32
BatchNormalization	
Activation	ReLu
Conv1D	64
BatchNormalization	
Activation	ReLu
Cond1D	128
BatchNormalization	
Activation	ReLu
GRU	128
Flatten	
Dense	Sigmoid

TABLE 2: Scheme of the proposed RNN with the layers and the correspondent learning parameters.

- **Afib dataset**: [1: 0.55 0: 5.63]
- **Normal dataset**: [1: 1.44 0: 0.77]
- **Other dataset**: [1: 4.84 0: 0.56]

We then applied them to the losses obtained with the *binary crossentropy*, by multiplying the given loss with the correspondent weight.

C. Cascade classifier ♣♠

The final classification was performed using a cascade classifier which employed the RNN trained with the three different datasets (presented in Section III).
The classifier is presented in Fig. 3 and works as follows:

1. Classifies Afib against all the other classes and assigns these samples the Afib label;
2. Discards all samples classified as Afib;
3. Classifies Normal against the Other and Noisy classes and assigns these samples the Normal label;
4. Discards all samples classified as Normal;
5. Classifies Other against Noisy class and assigns the corresponding labels;

VI. Results

A. Performance metrics ♣♠

To evaluate the quality of our models we used the following metrics:

- **Accuracy**: fraction of correct predictions;
- **Precision**: the ratio
$$\frac{Tp}{Tp + Fp} \tag{2}$$
where Tp is the number of true positives and Fp the number of false positives. It represents the fraction of ECG classified with a certain label that actually have that label;
- **Recall**: the ratio
$$\frac{Tp}{Tp + Fn} \tag{3}$$
where Tp is the number of true positives and Fn the number of false negatives. It represents the fraction of samples of a given label that are correctly classified with that label;
- **Fscore**: harmonic mean of Precision and Recall.

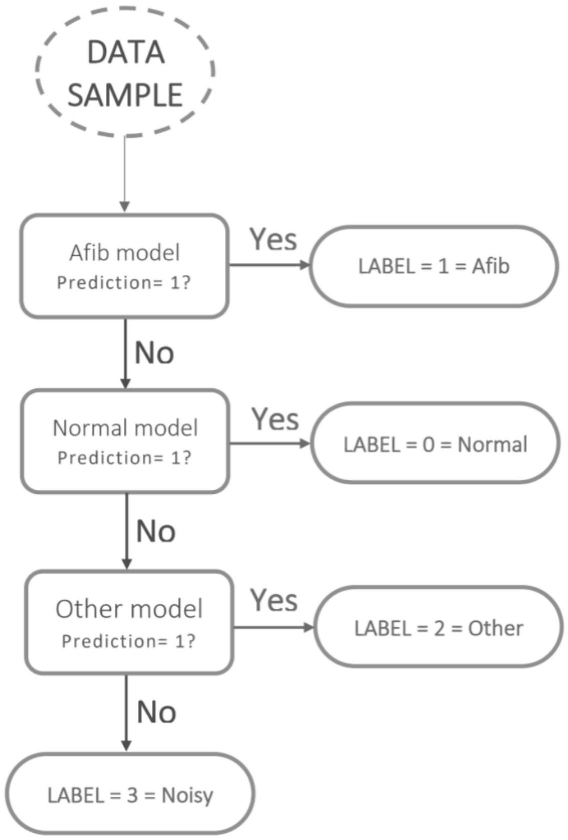

Fig. 3: Final classifiers hierarchy.

In order to compare our results with the other participants of the *2017 PhysioNet challenge*, we adopted the same F1 metric, defined as:
$$\frac{Fa + Fn + Fo}{3} \tag{4}$$
where Fa, Fn and Fo are the Fscore of the Afib, Normal and Other classes respectively.

B. Model without weights ♣

The validation performances of the model trained without the weights, taken to check the quality of the results before the final classification, are reported in TABLE 3.

Model	Accuracy	Precision	Recall	Fscore
Afib	95.16%	72.00%	74.38%	73.17%
Normal	84.24%	85.58%	91.27%	88.33%
Other	92.34%	94.91%	96.63%	95.76%

TABLE 3: Validation performances of the single trained models.

The final accuracy of our project is 81.18% with a F1 score of 0.77.

To compare the two model with and without our custom loss, we provide the Receiver Operating Characteristic (ROC) and Precision-Recall Curve (PRC): the ROC shows the false positive rate versus the true positive rate, the PRC, as the name suggest, illustrates the Precision over the Recall. In Fig. 4 and Figure 5 we reported the obtained plots for the models without the weighted loss function.

C. Model with weights ♠

In Table 4 we report the results of the classification (for the validation dataset) obtained with the models trained using the custom loss function, with the weights applied.

Model	Accuracy	Precision	Recall	Fscore
Afib	95.53%	77.78%	69.42%	73.36%
Normal	82.88%	85.55%	88.81%	87.15%
Other	92.58%	94.25%	97.67%	95.93%

TABLE 4: Validation performances of the single trained model with the custom loss function.

We obtained a final accuracy of 80.01% with a F1 score of 0.76.
We provided the ROC and PRC curve also for these models, which are shown in Fig. 6 and 7.

D. Results discussion ♣♠

As we can see from the numerical results shown in the the two previous paragraphs, we can say there was not an improvement derived from the use of the weighted loss function. Instead, the results were slightly better with the binary cross-entropy loss function without the weights.
Regarding the graphs in Fig. 4, 5, 6 and 7, we can compare the ROC and PRC curves both for the unweighted and the weighted loss function models.
The worst results for the ROC curve were obtained with the Normal classifier, with an area under the curve of 0.89 for the unweighted model and 0.88 for the weighted one; the best result was obtained for the Afib model, both for the unweighted and weighted ones.
For the PRC curve, the best results were obtained with the model that classifies the Other samples, while the worst performances are the one of the Afib model, both for the unweighted and the weighted one.
In general, we can say that weighted and unweighted models have a common behaviour, without strong differenced from one another.
The model which, in our opinion, needs to be enhanced in performances to improve the overall results is the Afib one; it represents the model that has to deal with the highest differences in terms of classes to label, so the performances are not at the same level of the other two, since they have to classify samples with less variability.

VII. Concluding Remarks ♣♠

In this paper we presented the multiclass ECG signals classifier for Afib, Normal, Other and Noisy rhythms. After the initial preprocessing of the signals, we divided the samples in multiple datasets to feed a RNN with them, in order to create a hierarchical classifier, in a *One-vs-All* strategy.
We presented the results we obtained, both using a binary cross-entropy loss function and a weighted version of it. The unweighted model was slightly better and our best F1 score was 0.77 with an Accuracy of 81.18%.
Even if we consider our model a simple and intuitive one with respect to other works presented for the *PhysioNet challenge* [5] we compared our result with the others and we managed to achieve the 7^{th} best result in terms of F1 rounded score.
Many things can be improved, starting from the data augmentation to cope with the unbalanced datasets, the addition of features extracted from the ECG signals, which may lead to a more complex model but possibly more accurate, and, finally, the number of epochs and the complexity of the layers of the model, which may be increased to enhance performances.
A final observation related to the use of the weighted loss function: we expected to obtain significantly better results and to be able to cope with the unbalanced dataset in this way, but the final results proved us wrong.

Project notes ♣♠

Getting comfortable with *Python*, *Tensorflow* and *Keras* while developing our project was not too hard thanks to the documentations and various internet blogs, but it still took us some time to write down our ideas. We would not say the developing environment was too hard to understand thought.
Our original project was very different from the one we presented: it was meant to work with the spectrograms of the signals rather than their time series. Firstly we wrote the code to transform all the signal in the frequency domain and we developed some methods for data augmentation, such as time warping and frequency masking, with a technique called *SpecAugment* [9]. We did not find many papers about it, as it was a rather new method mainly used for audio files, but we tried to adapt the idea to our project and we managed to produce the dataset to use. The problem was that no matter which RNN or CNN model we would use, the classification results converged to the same percentage of presence of the majority class (e.g. for the Normal, the accuracy was 59%, same as its presence in the dataset), because it classified all the samples as samples of that majority class, thus making a useless classifier. This last problem was hard to understand and we were not able to fix it. We tried to create our custom loss with weights such that the disparity of classes would be equalized, but we did not have the hoped results. Then we decided to extract some features from the signals and use them as aid for the classification: using the *wfdb* library we extracted ten features from every signal, such as the number of QRS complexes, the mean of the spectrum, the max of the amplitude and so on. After some trouble understanding how to create a CNN/RNN model with two inputs and one output in *Keras*, the results were the same as before.
It took us three full weeks of work, but we decided to not present this project and start over with what became our final paper, which is simpler, but works better; philosophically, as the Occam's razor teaches us, sometimes the simplest solution

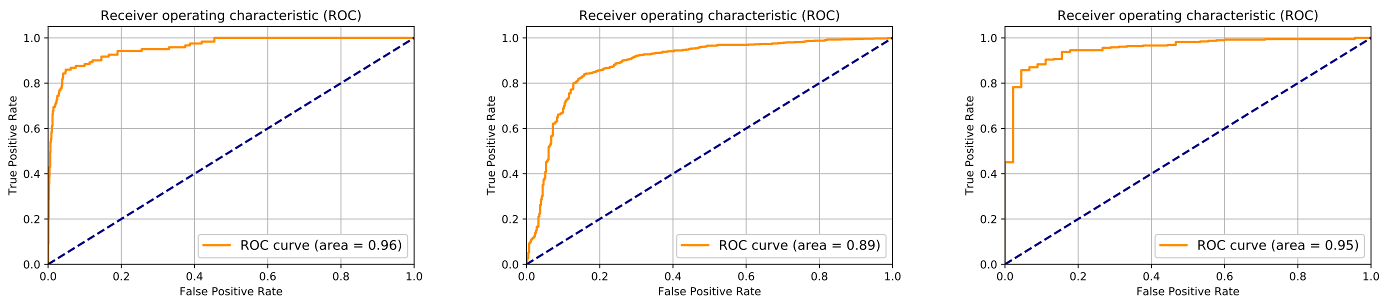

Fig. 4: From left to right: ROC of Afib classification without weights, ROC of Normal classification without weights, ROC of Other classification without weights.

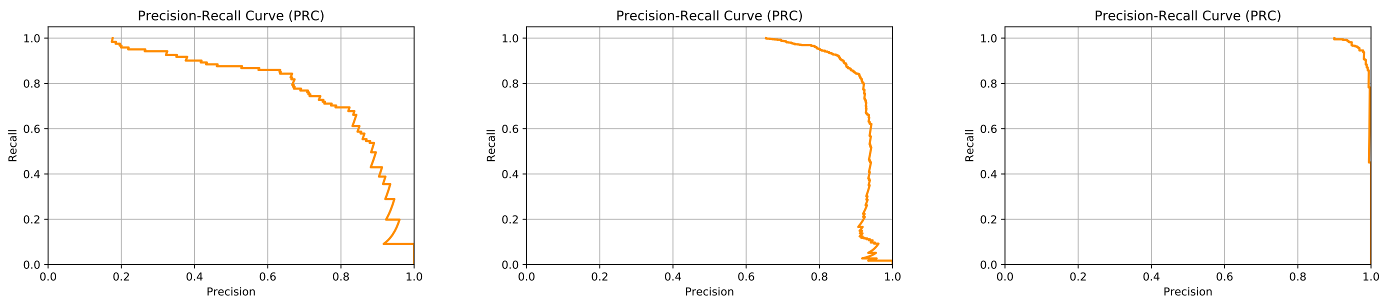

Fig. 5: From left to right: PRC of Afib classification without weights, PRC of Normal classification without weights, PRC of Other classification without weights.

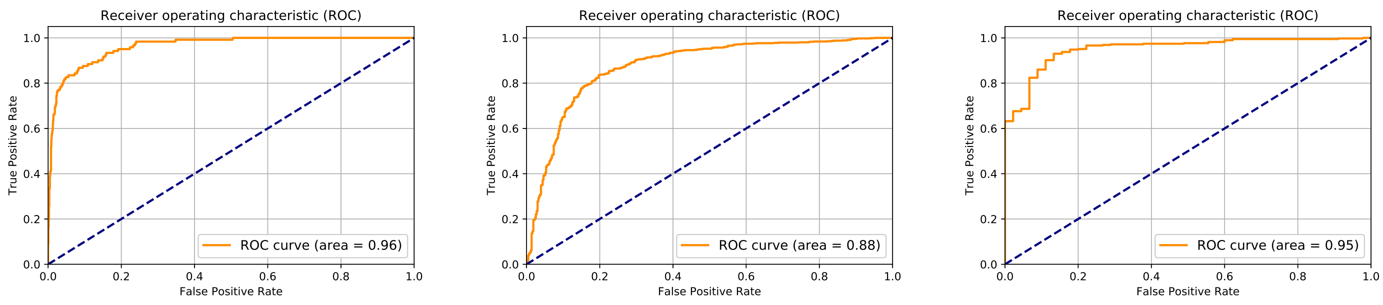

Fig. 6: From left to right: ROC of Afib classification with weights, ROC of Normal classification with weights, ROC of Other classification with weights.

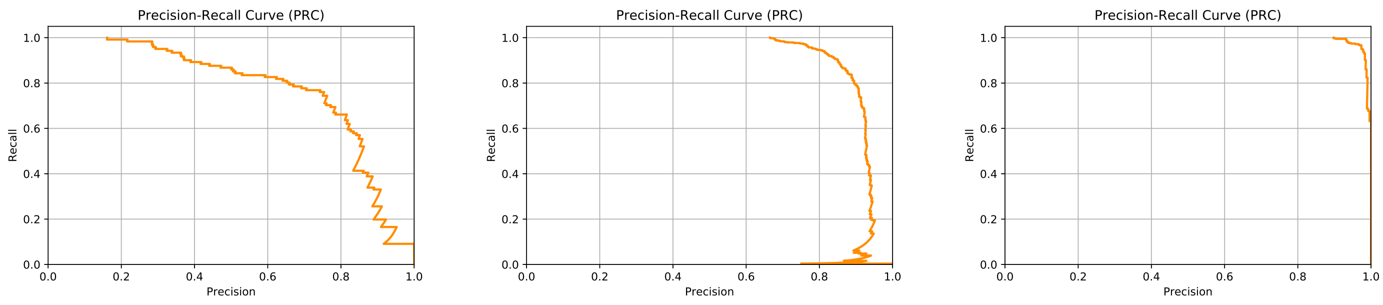

Fig. 7: From left to right: PRC of Afib classification with weights, PRC of Normal classification with weights, PRC of Other classification with weights.

is likely to be the best one.

The last difficulty we dealt with, which is not properly a problem, is the time needed to create the cache and then to train the model, but this is just related to out PCs as we are limited by the technology of our time.

REFERENCES

[1] S. Nattel, “New ideas about atrial fibrillation 50 years on,” *Nature*, vol. 415, no. 6868, p. 219, 2002.

[2] A. S. Go, E. M. Hylek, K. A. Phillips, Y. Chang, L. E. Henault, J. V. Selby, and D. E. Singer, “Prevalence of diagnosed atrial fibrillation in adults: national implications for rhythm management and stroke prevention: the anticoagulation and risk factors in atrial fibrillation (atria) study,” *Jama*, vol. 285, no. 18, pp. 2370–2375, 2001.

[3] J. A. Behar, A. A. Rosenberg, Y. Yaniv, and J. Oster, “Rhythm and quality classification from short ecgs recorded using a mobile device,” in *2017 Computing in Cardiology (CinC)*, pp. 1–4, IEEE, 2017.

[4] F. Andreotti, O. Carr, M. A. Pimentel, A. Mahdi, and M. De Vos, “Comparing feature-based classifiers and convolutional neural networks to detect arrhythmia from short segments of ecg,” in *2017 Computing in Cardiology (CinC)*, pp. 1–4, IEEE, 2017.

[5] G. D. Clifford, C. Liu, B. Moody, L.-w. H. Lehman, I. Silva, Q. Li, E. Johnson, and R. G. Mark, “AF Classification from a short single lead ECG recording : the PhysioNet / Computing in Cardiology Challenge 2017,” 2017.

[6] P. Warrick and M. Nabhan Homsi, “Cardiac Arrhythmia Detection from ECG Combining Convolutional and Long Short-Term Memory Networks,” vol. 44, pp. 1–4, sep 2017.

[7] M. Zihlmann, D. Perekrestenko, and M. Tschannen, “Convolutional Recurrent Neural Networks for Electrocardiogram Classification,” vol. 44, pp. 1–4, sep 2017.

[8] S. Parvaneh, J. Rubin, R. Asif, B. Conroy, and S. Babaeizadeh, “Densely Connected Convolutional Networks and Signal Quality Analysis to Detect Atrial Fibrillation Using Short Single-Lead ECG Recordings,” vol. 44, pp. 2–5, sep 2017.

[9] D. S. Park, W. Chan, Y. Zhang, C.-c. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” Google AI blog, apr 2019.

[10] M. Limam and F. Precioso, “AF Detection and ECG Classification based on Convolutional Recurrent Neural Network,” vol. 44, pp. 1–4, sep 2017.