Premature Ventricular Complexes (PVCs) and Premature Atrial Complexes (PACs) detector using an ECG-based Deep Learning approach

Course of Applied AI in Biomedicine, Politecnico di Milano

Giovanni Dispoto, Francesco Romanò

A.Y. 2021/2022

1 Introduction

Electrocardiograms (ECGs) are signals that record the electrical activity of the heart by use of electrodes that are positioned in the torso and limbs of the patient. The signal is composed of several leads that are related to the heart activity from a particular angle, acquired by electrodes in various parts of the body. After the acquisition of the ECGs, it may be interesting in discriminating if a beat is a Normal Sinusoid Beat or not normal. Several pathologies could affect the patient and in particular, we are interested in Premature Ventricular Complexes (PVCs) and Premature Atrial Complexes (PACs).

In recent years, several approaches based on Machine and Deep Learning were proposed to tackle this task. It is important to underline that we can distinguish two tasks: Intra-Patient (or patient-specific) beat classification and Inter-Patient beat classification. The Intra-Patient task consists into train a model that can predict a class of a beat of one or a few patients. The Inter-Patient task instead, consist into train a model that can predict the class of a beat of different (and new) patients.

As it is possible to understand, while the first task is easy to tackle, the second one is a more general and challenging task due to the difference of the patients. To tackle the Inter-Patient task, it is important not to have a beat from the same patient in the training set, validation set or test set.

In this project, we propose a 1-D Convolutional Neural Network (1D-CNN) to classify a beat as Normal Sinusoid Beat (N), Premature Ventricular Complexes (S) and Premature Atrial Complexes (V). Our model relies completely on the ECG signal, without explicit use of features. Our model reached an accuracy of 95% on N, 83% on V and 76% on S on the test set.

Section 2 describes the dataset and the pipeline developed to tackle the task. Section 3 shows the results of our experiments. Section 4 discusses the results obtained by experiments and Section 5 draws the conclusions and possible future works.

2 Materials and Methods

We present our pipeline in a high level way in Figure [1], all the steps will be then described in detail. Subsection 2.1 describes the dataset used to train the models. Subsection 2.2 describes the preprocessing done on the ECG to clean the signal. Subsection 2.3 describes the resampling technique used to mitigate the class imbalance issue. Subsection 2.4 describes the model we used. Subsection 2.5 describes the model selection and validation procedure. Subsection 2.6 describes some explainability techniques we applied to give additional interpretability of results.

2.1 Dataset

The dataset provided to solve the task is composed of records obtained from 105 different patients. Each record is characterized by three files: the ECG 2-leads signal of the patient (with a variable length between patients), the labels of the beat class of all the record beats ('N' for normal beats, 'S' for PACs, and 'V' for PVCs) and the positions of the R peaks of each beat. ECG recordings were not sampled at the same

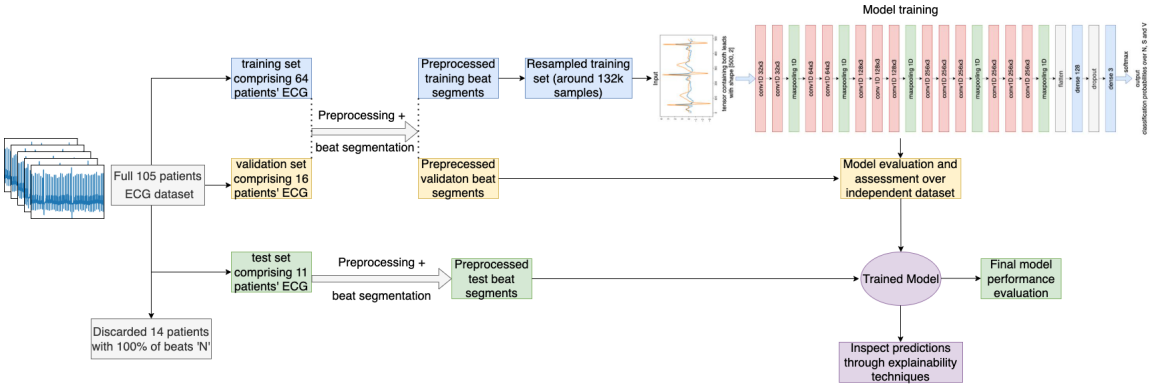


Figure 1: High level view of our pipeline (the model diagram is showed more clearly later)

frequency, indeed some patients' ECG was sampled at 128 Hz while others were at 250 Hz. The dataset contains lots of Normal (N) beats since only a small portion of them are anomalous. This leads to a strong imbalance between the classes of the task that needs to be handled properly as it is possible to see in Figure [2]. To implement correctly a Deep Learning pipeline, the dataset must be split into a training set, a

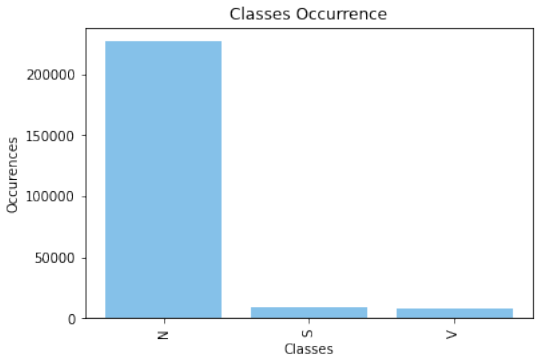


Figure 2: Distribution of the labels in the dataset

validation set used to perform model selection and a test set used for final evaluation. The most important aspect that must be considered is that to obtain a model able to generalize over patients never seen by the trained model, we have to create sets that do not share beats of the same patients. This increases a lot the task difficulty since by a simple exploration of the data it is evident that different patients' recordings show different shapes, probably due to various recording settings, instrumentation, human errors in the process. Another important aspect to consider is the distribution of the beats classes of the various patients: to have good training we want to feed the model with all kinds of beats and validate it on a validation set with a similar classes distribution. Taking into account these important assumptions we have manually selected the patients to create a good split. More precisely we selected 64 patients for the training set, 16 for the validation set, 11 for the test set with a similar class distribution and we discarded the remaining 14 patients because they were composed entirely by 'N' beats and we suppose they were not useful for the classification task since they cannot give information about other classes.

2.2 Data Preprocessing

After the initial split of the data, a correct preprocessing of the ECG signals must prepare them for the model.
First of all, we considered the two different sampling frequencies used. To solve this problem and to obtain the same frequency over all the data we resampled all signals to 250 Hz. The effect of this is visible in the figure [3]. After this step, we obtained a coherent behaviour in intervals of fixed length over all the patients. Then we took into consideration the noise over the recordings, often present in this type of data. In particular, the signals were affected by a high-frequency noise, which consists in small disturbances along with the signal, and a low-frequency noise, which consists in a drift visible only if we look at long chunks of the signal and that changes the baseline point of various beats. We then performed a denoising procedure inspired by the referenced paper [GMC21]. It consists in the application of two median filters of 200 and 600 ms length to obtain a baseline wander estimate to be removed from the signal and obtain a baseline-corrected one, and

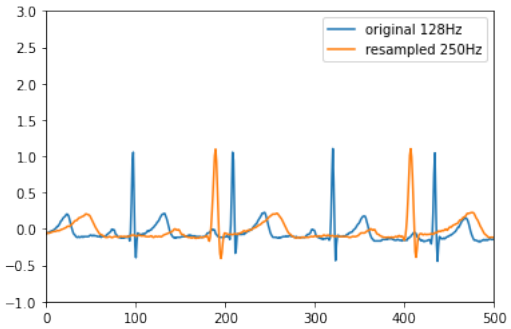
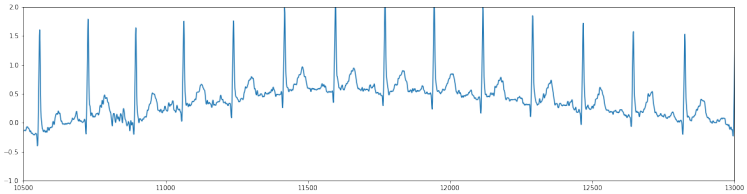
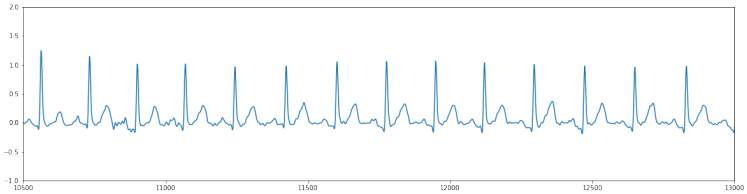


Figure 3: Example of the effect of the resampling on a piece of ECG

then a finite impulse response (FIR) low pass filter with a cutoff of 35 Hz to remove the high-frequency noise. It is very clear to see the effect on the figures [4a][4b]. Once the signals were cleaned, considering



(a) ECG signal before preprocessing



(b) ECG signal after preprocessing with two median filters of 200 ms and 600 ms and a FIR low pass filter with cutoff of 35hz

Figure 4

the beat classification task, we performed beat segmentation to extract single heartbeats from the ECG: these segments are the input of our model. We used a window of fixed length centered in the R peak of the considered beat to obtain them, in particular we used a 500 samples window that on average contains the interested beat, but also the peaks of previous and next ones. We think that giving some context of the signal near the beat is important to allow the model in extracting some high level features useful to identify some abnormalities like short RR distances between subsequent beats. In particular, the heartbeat context seems particularly useful to identify S beats, result that is in line with the theoretical aspect of PVC beats. In the end, we applied z-score normalization over our data: this step is very important since it reduces meaningless differences between data and allows the model to better focus on differences meaningful for the classification task.
The previous procedure was applied to both the two channels (leads) of the given ECG and the single input of our model was a tensor of shape (500, 2), that consists in the fixed window segment of each channel. In the figure [5] it is possible to see an example of a z-scored heartbeat.

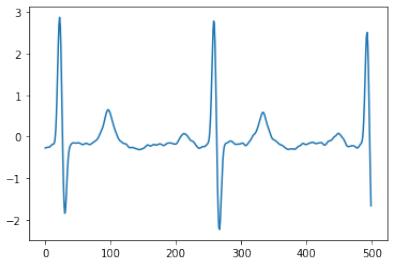
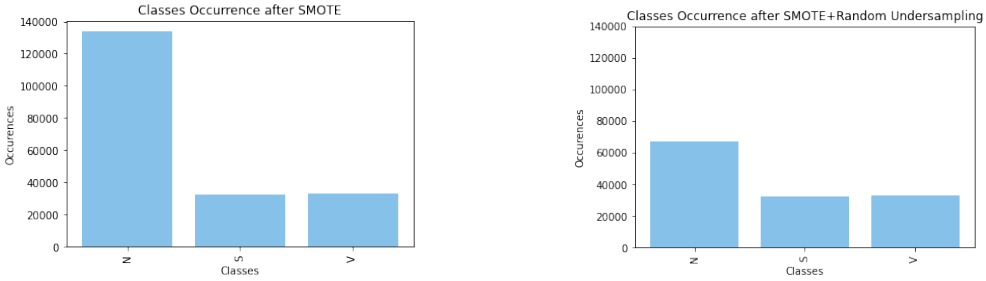


Figure 5: An example of a heartbeat after all the preprocessing steps (only one lead is shown for clarity)

2.3 Resampling

As stated before, the dataset suffers from a high class imbalance. Our solution to this problem, which could bring a strong bias to the model, consisted into performing a resampling of the dataset. Resampling is a way to balance the class distribution in the dataset, both by generating new samples of minority classes (oversampling) and by discarding samples of the majority class (undersampling). In particular, we used Synthetic Minority Oversampling TEchnique [Cha+02] (SMOTE) to make oversampling of S and V data points. SMOTE algorithm generates synthetic data points of the desired class by considering the 5 nearest neighbours of the minority class data point (point a), picking at random one of them (point b) and making a convex combination between the two points a and b (the results are visible in the figure [6a]). In combination with SMOTE, we have also applied Random Under Sampler to reduce the number of the N data points, this last algorithm simply discards at random data points of the desired class (the results are visible in the figure [6b]). At the end of our resampling procedure, we obtained a more balanced class distribution, mitigating the initial problem.



(a) Distribution after SMOTE

(b) Distribution after SMOTE + Undersampling

Figure 6

2.4 Deep Learning Model

To solve the non-trivial classification task, we decided to use a Convolutional Neural Network (CNN) which is able to automatically perform features extraction from the preprocessed ECG signals. Since our data is a mono-dimensional vector, we used a 1D-CNN, using 1D convolutional layers as it is possible to see in Figure [7]. The convolutional part of the network works as a features extractor since convolutional filters are able to capture basic signal features in the first layers and then more and more high-level features learned to be useful to our classification task in deeper layers which has a bigger receptive field over the input and can identify high-level patterns. As usually done in these kinds of models, convolutional layers are interleaved by Max Pooling layers to reduce gradually the size of the input.
The model proposed is inspired by the famous VGG model [SZ14], with only small changes. It presents multiple convolutions in sequence, but with a small filter size (3x3), to achieve large receptive fields with fewer parameters and more non-linearities than models with larger filters in a single layer. After this features extractor part, the network ends with a Flatten layer which orders the features in a vector ready to be processed by the final Fully Connected classifier, composed of simple Dense layers. The final output of the network is composed of 3 neurons, that will store the predicted probability estimated by the model for each heartbeat class, thanks to the typical 'softmax' activation function.

The network was then trained by minimizing the *Categorical Cross-entropy* loss function, typical for multiclass tasks, and the Adam optimizer with a very low learning rate (1e-6) to improve learning stability. Data was fed in the model in a mini-batch way, giving at each learning step a mini-batch of 32 input heartbeats. The model validation loss converged in around 20 epochs.

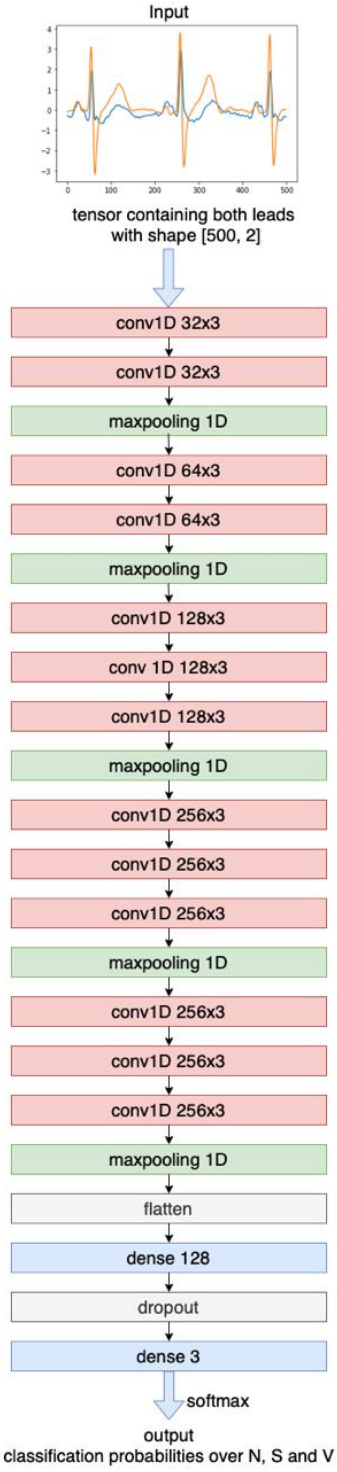


Figure 7: 1D-CNN model proposed

2.5 Performance Metrics

The metrics that we used in order to evaluate the performances are precision, recall and F1-score defined as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{Total}$$

These metrics are really important since some other metrics like a generic loss or accuracy are biased by the data distribution. For instance, in our case, a model that predicts every time N would reach an accuracy $\geq$ 90 %, so it is important to rely on more meaningful metrics like precision, recall and f1-score.

2.6 Model Assessment

We performed several training changing model layers and hyperparameters, and we used two main approaches to perform model selection.
Firstly, we weighted the validation loss to be more sensible in improvements on the classification of classes with fewer samples in the validation set. This allowed us to have a less biased estimate of the overall behaviour of the model also during the training.
Secondly, we monitored classification metrics for each class and not for the average over all classes. This allowed us to directly see the performances related to each class during the training, and we could identify cases in which the model seemed good by looking only at the averaged metrics but it performed well only for a single class for instance.
After the training, we checked the results by plotting the confusion matrix that shows more clearly all the predictions done by the model, and we evaluate it especially by looking at the precision, recall and f1-score of each class that summarizes well the quality of the classification, but also the overall accuracy and the macro-average were considered.
After the validation step, we performed a final evaluation using the heartbeats of the test set. We remark that to have a good idea of the generalization power of the model, both validation and test set contains heartbeats obtained from patients never seen during training. In Section 3 we show in detail our best model results both on the validation set and on the test set.

2.7 Model Explainability

Here we want to focus on the explainability of our CNN model. It is typically known that Deep learning models are hard to explain and often results are taken in a black-box way. To mitigate this problem, we implemented two different explainability techniques: GradCAM and LIME.

GradCAM is a technique able to give an explainability heatmap extracted from the last convolutional layer of the CNN, the layer which extracts the final features vector and maintains the spatial information which is then lost in the fully-connected layers. The heatmap, in the case of 1D-CNN, will be a vector of length equal to the size of the last convolutional layer output (in our case 31) and each element of the heatmap is a number that represents how much the model is focusing on that part of the input to perform the classification. Obviously, this technique gives a coarse explanation of the model since the heatmap is much shorter with respect to the original input size (in our case 500), anyway, we think it is a first step to understand what the model is looking for to classify the samples.
As explained in the original paper [Sel+16], the heatmap is obtained exploiting the gradient flowing information of last convolutional layer to assign importance values to each neuron for a particular decision of interest. We first compute the gradient of the score for class c, y^c, with respect to feature map activations A_k and we average these gradients over the lenght of the layer output Z to obtain the neuron importance weights $\alpha_k{}^c$:

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

In figure [8] we show the visualization of the heatmap over the original input sample taken from the validation set for each heartbeat class (only one of the 2 leads is shown for clarity):

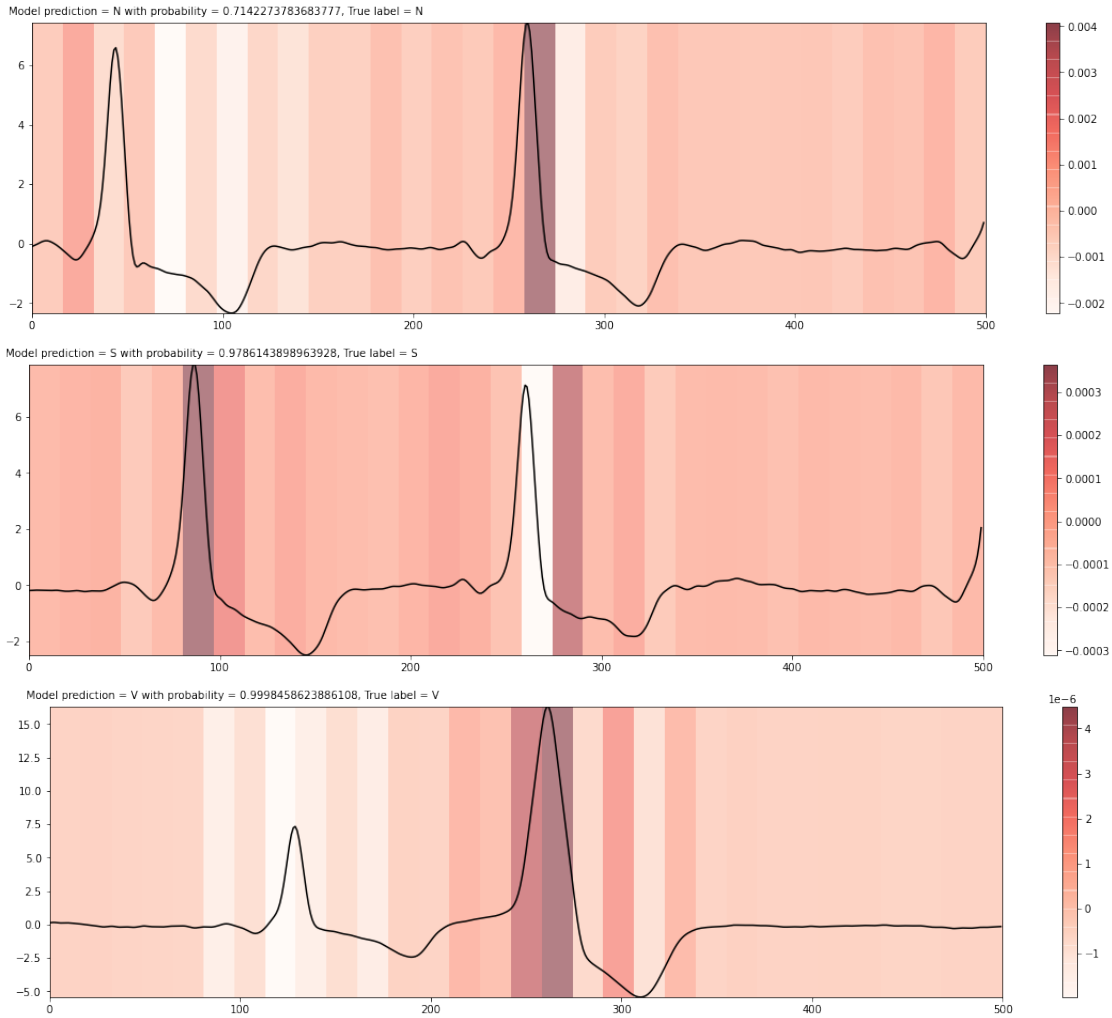


Figure 8: GradCAM visualization over 3 examples taken from validation set explainations

LIME (Local Interpretable Model-Agnostic Explanations) is a model agnostic instance-based explainability technique. The idea is to give to the algorithm a single data sample, then LIME perturbs that sample, pass to our CNN model all the 5000 perturbed data points and collects the model predictions probabilities outputs. Finally, it builds a simple linear model which approximates the complex CNN in a small local space that is easily explainable, using as a fictitious dataset the 5000 perturbated points and as labels the model predictions. In the end, LIME provides the importance of every single element of the sample used (that are considered as the features of the simplified local model), and this assures a more fine-grained explainability visualization.

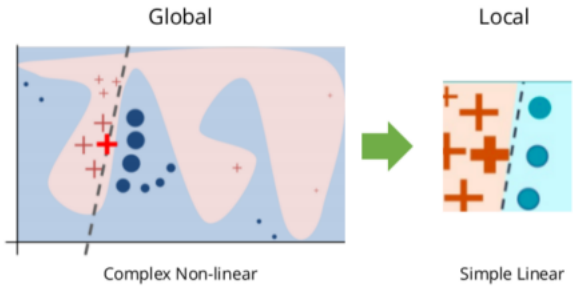


Figure 9: Image taken from original paper [RSG16] that shows the basic idea behind LIME

In figure [10] we show the explainations visualized for the same examples used for GradCAM to compare the two methods:

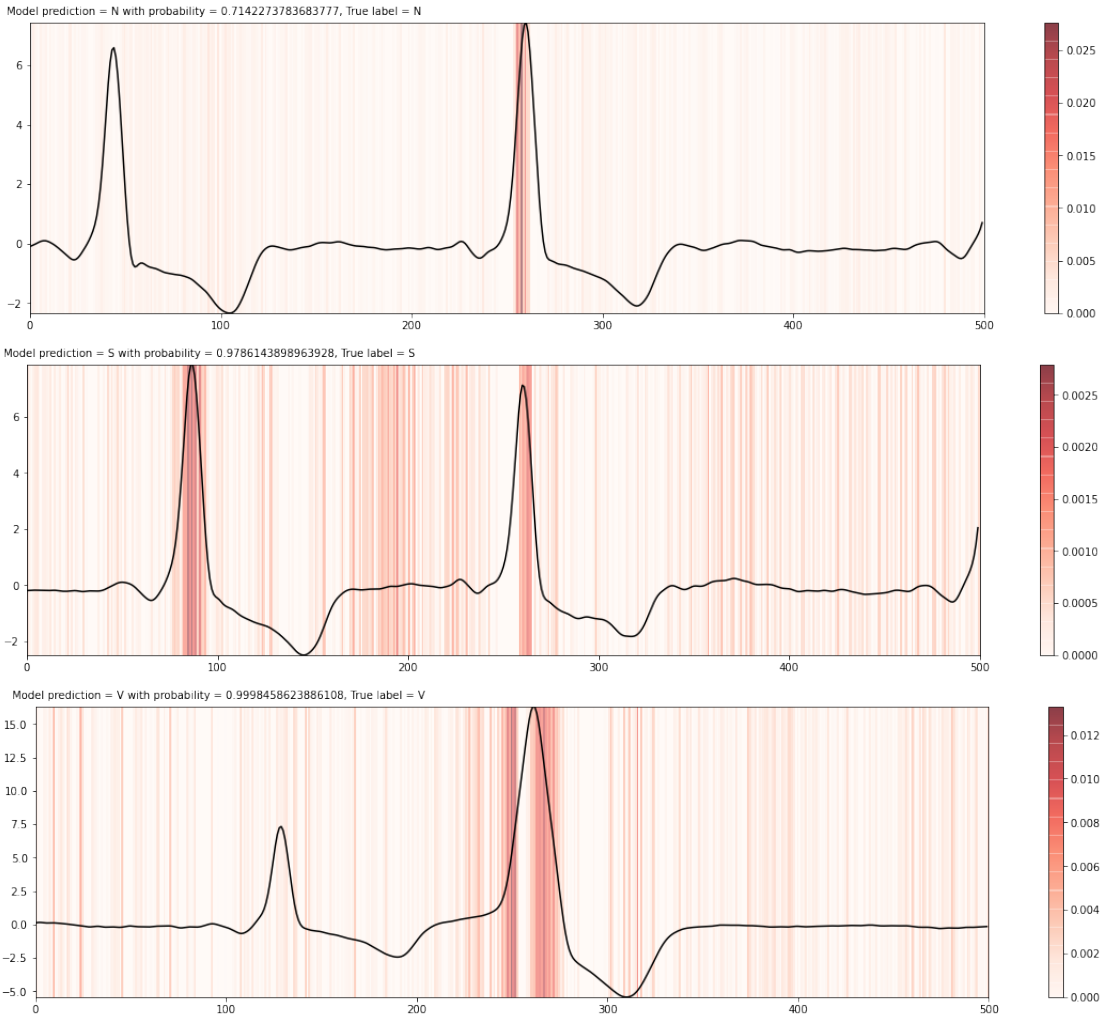


Figure 10: LIME visualization over the same examples used in figure [8]

As can be seen, we show also the prediction probability and the real label, to furtherly investigate the results. We can appreciate similarities in the explanations between the two methods: for instance, the S beat appears explained through its nearness to the previous beat and a small region near to its peak, while the V appears explained only by its peak region. It can also be useful to use these techniques to inspect misclassified samples and understand why the model has failed the classification of the beat.

3 Results

Once obtained the best mode, we tried to tune some hyperparameters values and optimizers to see if there could be an impact on the performance. In particular, in the table [1] we compare the performance of the model with Adam, Adadelta, Nadam and SGD with momentum (momentum fixed to 0.3). Since Momentum and Adadelta required a higher learning rate w.r. to the other two, we set it to an arbitrary chosen higher learning rate. The best optimizer according to the F1-score macro is Adam.

Optimizer	Learning Rate	F1-score (N)	F1-score (S)	F1-score (V)	F1-score macro
Adam	1e-6	0.97	0.60	0.65	0.75
Momentum	1e-2	0.96	0.68	0.52	0.72
Nadam	1e-6	0.95	0.51	0.46	0.64
Adadelta	1e-2	0.96	0.64	0.49	0.70

Table 1: F1-performances reached by various optimizers

Moreover, we tried to vary the number of neurons in the dense layer to see if there are some improvements on the validation set, as it is possible to see in table [2]. As it is possible to see, the optimal number of neurons in the Dense layer is 128.

#neurons in Dense	F1-score (N)	F1-score (S)	F1-score (V)	F1-score macro
32	0.97	0.54	0.62	0.71
64	0.95	0.54	0.50	0.66
128	0.97	0.60	0.65	0.75
256	0.94	0.60	0.36	0.63

Table 2: F1-performances reached by various number of neurons in dense layer

In the figure [11] it is possible to see the loss of the best model obtained. The validation loss is much higher since we are using a weighted one, but the training curve is quite smooth. In figures [12] it is possible to see the confusion matrix on the validation set. The model reached quite good performances with good accuracy.

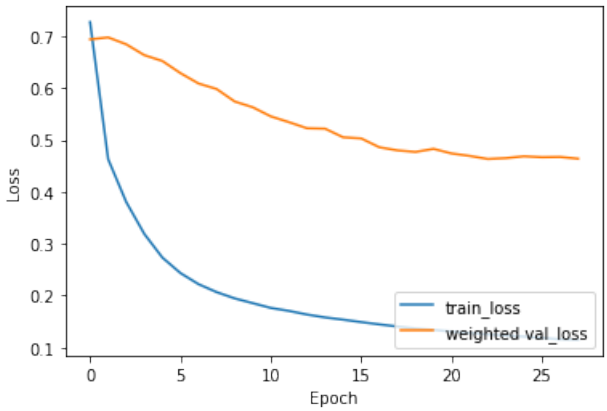


Figure 11: Training and weighted validation loss

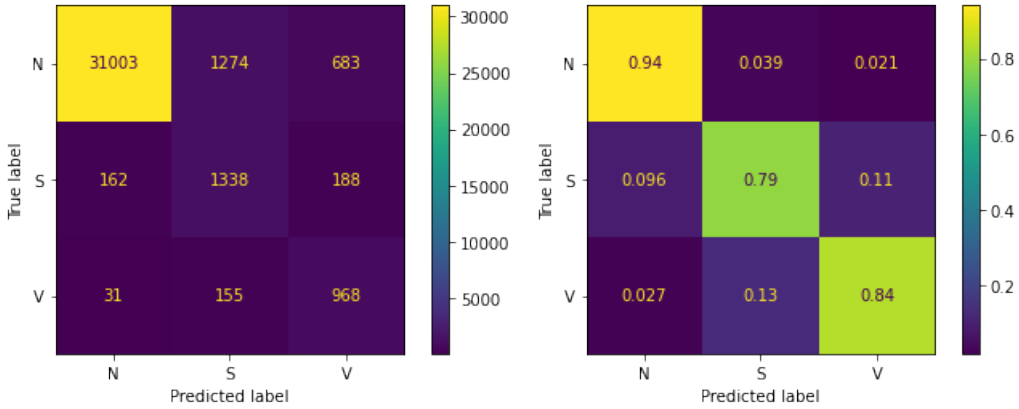


Figure 12: Confusion Matrix of the validation set

In the figure [13] it is possible to see the confusion matrix on the test set. In this case, the accuracy on the S beats increase a little bit, but with a 0.58 f1-score. On the V beats instead, the accuracy decreased slightly but with a 0.77 f1-score. The overall performances on the test set are available in the table [3].

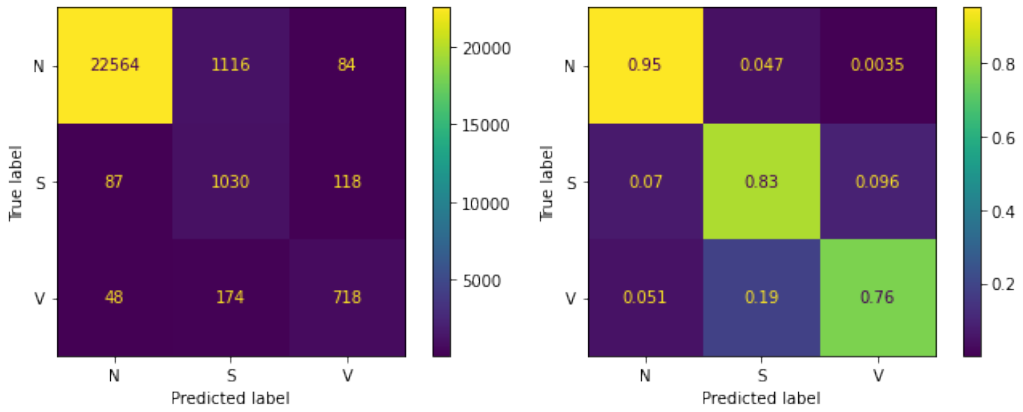


Figure 13: Confusion Matrix on the test set

Class	Precision	Recall	F1-score
N	0.99	0.95	0.97
S	0.44	0.83	0.58
V	0.78	0.76	0.77

Table 3: Performance of the proposed model on the test set

4 Discussion

The model proposed is able to detect PVC and PAC beats with an accuracy of at least 0.75 in the worst case (V) on unseen patients. This is a good result since many articles in the literature focus their attention on the intra-patient task, by validating the model on the very same patients used during the model selection phase, obtaining apparently higher performances due to the bias introduced into the test set. The main problem that needs to be fixed in the model proposed is the precision on the S beats that is quite low and this also affect the f1-score of the S class.

5 Conclusions

In this project, we proposed a 1D-CNN that is able to predict, in an inter-patient fashion, if a beat is Normal, PVC or PAC relying on only a portion of the ECG. In particular, the CNN requires in input a window length of 500 samples around the R-peak of the beat that we want to predict. This model gives us a quite good result with an accuracy ≥ 0.75 in the worst case related to the PVC class. Thanks to the explainability techniques implemented, we were able to understand that the model is exploiting the correct information from the input in order to predict the class, but also to try to understand why the model miss-classify some beats, giving us a powerful tool for results inspection. In the future, the proposed model could be improved by trying to increase the samples of the S class that is the most difficult one to learn.

References

[Cha+02] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority over-Sampling Technique". In: *J. Artif. Int. Res.* 16.1 (June 2002), pp. 321–357. ISSN: 1076-9757.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[RSG16] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[Sel+16] Ramprasaath R. Selvaraju et al. "Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization". In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391.

[GMC21] Guadalupe García-Isla, Luca Mainardi, and Valentina D. A. Corino. “A Detector for Premature Atrial and Ventricular Complexes”. In: *Frontiers in Physiology* 12 (2021). ISSN: 1664-042X. DOI: 10.3389/fphys.2021.678558. URL: https://www.frontiersin.org/article/10.3389/fphys.2021.678558.