Heartbeat classification using Machine and Deep learning techniques

Francesco Masciulli
DEIB
Politecnico di Milano
francesco.masciulli@mail.polimi.it

Francesca Pietrobon
Dipartimento di Matematica
Politecnico di Milano
francesca.pietrobon@mail.polimi.it

Simone Tomè
DEIB
Politecnico di Milano
simone.tome@mail.polimi.it

January 25, 2022

Abstract

In this project, some Machine Learning models have been applied for classifying three different types of beats in an ECG: normal sinus rhythm beats like , ventricular beats like premature ventricular complex beats (PVCs), and supraventricular beats like premature atrial complex beats (PACs). The methodology used was carried out applying as input of the models the preprocessed signals divided into patches centered on the peak to be classified. The models inspected are classical Machine Learning models such as Support Vector Classifier, K-Nearest Neighbors and Random Forest Classifier, and also complex Neural Networks like ResNet or CNN with LSTM. Those methods are trained not only with the preprocessed input but also with a reconstructed input obtained by an Autoencoder that constructs easily identified patches of beats.

1 Introduction

The usage of artificial intelligence in medical applications is gaining more interest in the latter decade. The task of interest is the classification of each heartbeat instance in an electrocardiogram signal (ECG). This aims to detect arrhythmias in the normal sinus rhythm and to classify such abnormalities. In this specific case, we are trying to distinguish between normal sinus rhythm beats, supraventricular beats, and ventricular beats. The supraventricular beats are premature atrial complex beats (PACs), instead, ventricular beats in most cases are premature ventricular complex beats (PVCs) but could be also other types of beats with a shape like the PCVs ones.
In normal sinus rhythm, the impulse that is generated in the sinus node is propagated through the atrium and then to the ventricle.
In PVCs instead, the impulse is generated in a random place in the ventricles so the contraction appears before the atrium and they are going to depolarize differently. In the ECGs of PVCs, we will have a premature beat and a distortion of the beats curve because the origin of the depolarization is

different and it is going to spread differently through the ventricles.
PACs instead are going to induce also a distortion of the rhythm because it is going to occur before the normal sinus rhythms impulse as in PVC, but we will have a morphological distortion depending on where the PAC is originated. If it is very close to the sinus node it is going to have very similar P waves, instead, if it is in another place we are going to have a different P wave, but it is a very slight morphological distortion since the P wave have a lower amplitude so it is easily influenced also by noise. Then we have the usual convex curve because at the end the impulse is going to be propagated normally to the atrioventricular node and through the ventricles so we don't have any distortion. For these reasons, it is more difficult to see the difference based on morphology.
So it is easy to study PVC because they have high amplitude but there is still a lacuna of detecting PAC. It is an open area of research because it is not possible to classify manually each peak due to the huge amount of data present in ECGs.
To address this task the work was divided into different sections. In section 2 the available data are analyzed and presented. The work goes on with section 3, where data manipulation and the proposed architectures are described. The assessment process of such models is defined at the end of the same section, clearly defining the steps used. The outcome of the experiments is analyzed in section 4 highlighting the different performance obtained by each classifier.

2 Materials

The dataset comprises 105 2-leads ECG signals from different patients, among with the location of the peaks that have to be classified. Each signal length is 30 minutes and 65 are sampled using a frequency of 128 Hz, while the remaining ones are sampled at 250 Hz. Being supervised learning each RR-peak of each QRS complex is annotated with the class label. Labels can be: N (normal), V (PVC), or S (PAC), and each label is associated with the RR peak position in the signal. In fig. 1 we can see the distribution of the average RR-peaks periods for each signal. The periods range from 0.95 seconds to 0.42 seconds (this translates into 63 Bpm to 143 Bpm) and this is independent from the sampling frequency. The distribution of the amplitudes is reported in fig. 2.

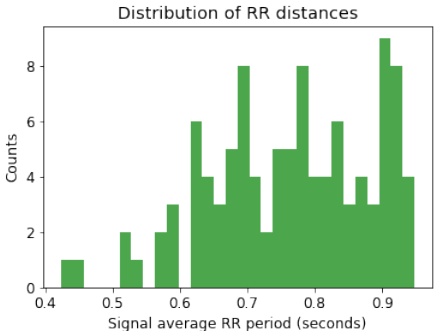

Figure 1: Distribution of RR periods. We define the period as the distance between two peaks of the QRS complex.

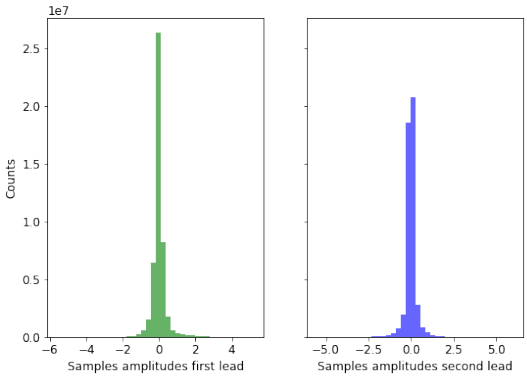

Figure 2: Amplitudes of the samples (considering all patients) for the first lead signal and second lead signals.

We can observe that the expected value of a sample is 0 for both leads, but we will see that many signals have trends that need to be removed. Those trends can be observed using the Fourier transform and considering very low frequency coefficients. For example, in signal of the patient labeled *S084* we have a frequency spectrum (just considering the first 16 seconds) that can be seen in fig. 4.

The noise present in the signals does not limit to low frequencies artifacts. Infact, also high-

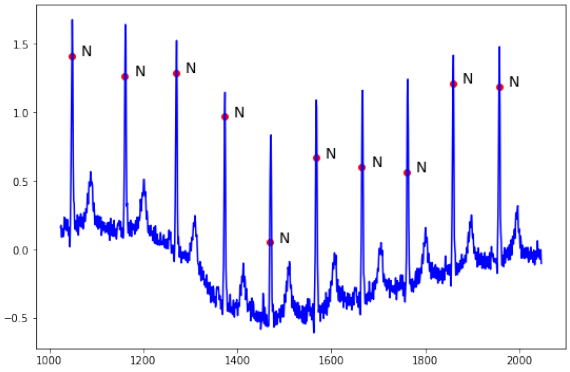

Figure 3: Signal for patient *S084* between 8 and 16 seconds. The baseline wander can be distinguished clearly. Those kind of artifacts may be due to breathing or moving of the patient and need to be addressed with filters.

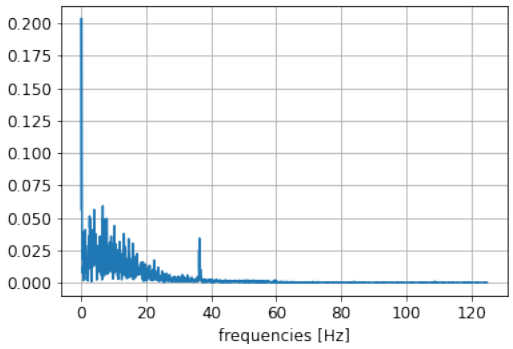

Figure 4: Fourier transform of the signal (first 16 seconds). A peak at 0.0625 Hz suggest the presence of a low frequency artifact: the baseline wander.

frequency noise is present and requires to be addressed properly with ad-hoc filtering techniques. High-frequency noise can be observed on the plots of the signal. Abrupt changes in the signal that gives a serrated aspect are caused by high-frequency noise. The plot in fig. 3 shows both kind of noise: the low-frequency one gives the trend to the signal, while the high-frequency one disrupts the smoothness of the sampling and sometimes introduces artifacts that may distort the QRS complexes, making the classification much harder as studied by Kher [6].

In ECG signals, even if arrhythmias are present, the most frequent beat is expected to be a normal one. This introduces another problem in the training: the class imbalance, a problem encountered also by Sarvan and Özkur [13]. The class imbalance may jeopardize the training, biasing the classifier towards normal beat instances (Japkowicz [4]). If the average signal contains at least 90% (fig. 5) of normal sinus rhythm beats and we build a classifier that classifies all instances to be a normal one, we will achieve 90% accuracy: this high accuracy is misleading and we need to address this with techniques discussed in section 3.

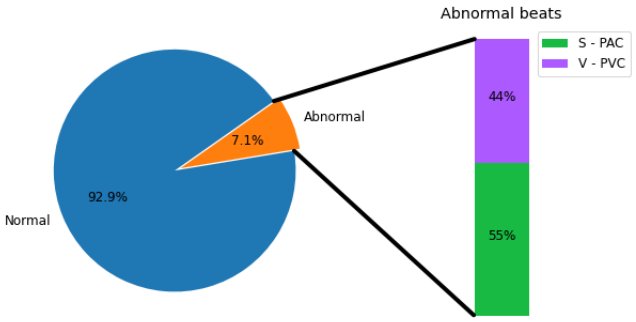

Figure 5: Distribution of the different classes.

3 Methods

3.1 Preprocessing

After the dataset creation, including each lead signal, the locations of the peaks, and their labels, the raw signals are initially resampled with a sampling frequency of 128Hz. This choice was necessary in order to have each signal with the same number of samples. Then they are filtered with a bandpass filter, cutting the components below 0.8 and above 45 Hz. The band is chosen from the reflections of Murthy et al. [9], but considering that PVC peaks are characterized by higher frequencies. The spectrogram of the signal before and after the filtering is shown in fig. 6 and fig. 7. The filter was implemented through a digital Butterworth filter of order 6, and the passing band was computed starting from the Nyquist frequency of the signals. The filtering process successfully removed the baseline wander and slightly smooth the signals, cutting the higher and lower frequencies. In fig. 8 is shown the result of preprocessing, comparing the same portion of signal *S084* shown in fig. 3. Finally, each filtered signal is scaled to a range [0, 1] considering time windows of 10 seconds. The scaling is inspired by preprocessing methods of Kachuee et al. [5] work, among with the following patch extraction.

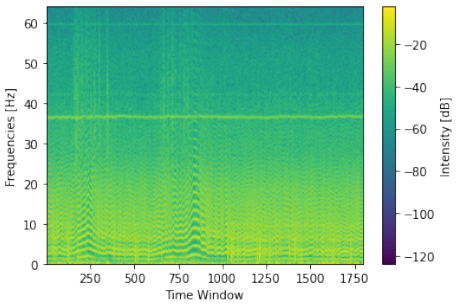

Figure 6: The spectrogram of *S084* raw signal

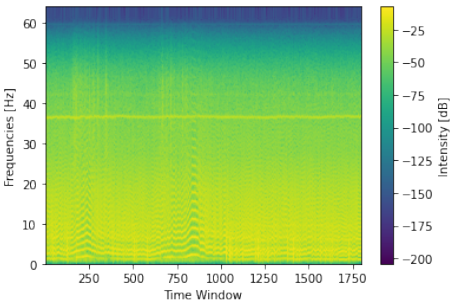

Figure 7: The spectrogram of *S084* signal after being filtered

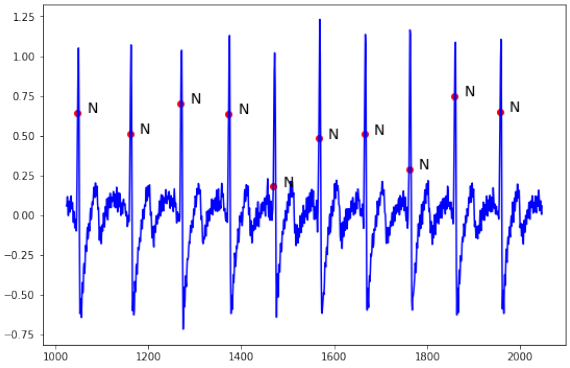

Figure 8: Signal for patient *S084* after applying the bandpass filter, showing the same time interval of fig. 3. Can be observed that through filtering the local trend of the signal was removed.

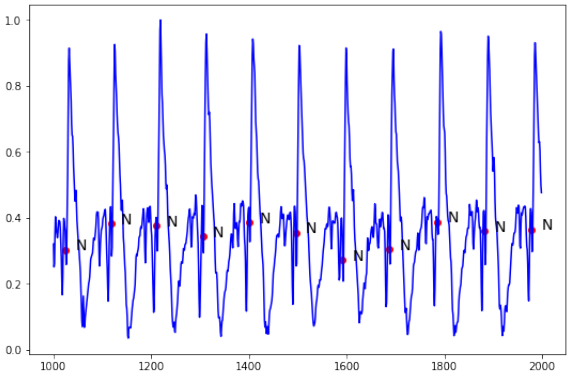

Figure 9: Signal for patient *S093* identified as outliers

3.2 Outliers removal

Also after the preprocessing, some ECG signals remain distorted (fig. 9), this could be caused by various factors that characterize each patient signal, such as respiration, movements, displacement of the electrodes or errors during the measurement acquirement. Those signals could be identified as outliers. Notice that such signals will not be used during neither model training nor evaluation. This choice is moved by the assumption that such outliers are likely not only samples which features are some way different from the others. Instead they are considered as signals that significantly differs from all the others, having passed the same preprocessing, due to the above-mentioned factors and, thus, not relevant to the task.

As we will see in section 3.3, signals will be analyzed in patches so to detect outliers it is necessary to treat every single patient as unique data. The outliers are detected computing as summary statistics mean, standard deviation, variance, skewness, and kurtosis for both the leads and to be able to compare them z-scoring is performed.

$$Z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean and σ the standard deviation.
Signals are discarded if $|Z| > 3$, in this way six outliers patients are discovered and not considered during training nor evaluation.

3.3 Patch extraction

In order to include the most representative information based on the different kinds of beats that had to be classified, a patch is extracted and fed as input of the models. Others have taken this approach to the task, such as what is done by Anwar et al.[1]. For each signal, a patch is taken out for each peak. The patch is centered on it and contains the following and preceding signal for a length, in both directions, equal to the average of the RR distances of the single signal [5]. In the case of border peaks, the patch is padded with a constant value (0.5 as it is the mean for the scaling interval). In the end, each patch is then resampled in order to have exactly 350 samples, due to the fact that each input of the model needs to be of a fixed dimension. It is important to highlight the fact that the dataset split is made before patch extraction, in order to ensure that the validation set will contain only the peaks belonging to patients that are not seen during training (see section 3.6). In the following (fig. 10, fig. 11, fig. 12) is shown an instance of, respectively, a normal, PAC and PVC beat.

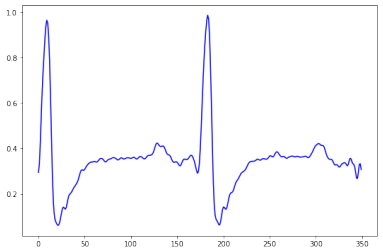

Figure 10: Extracted patch for an Normal peak.

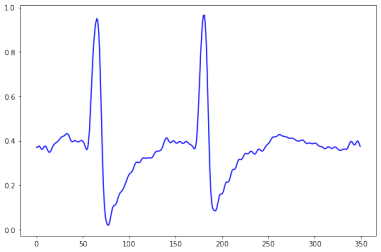

Figure 11: Extracted patch for a PAC peak.

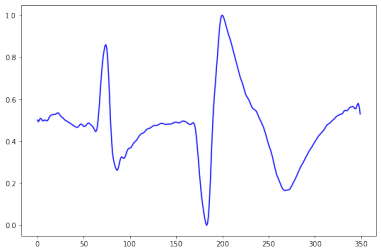

Figure 12: Extracted patch for a PVC peak.

3.4 Dataset balancement

One of the most used techniques to train a model with unbalanced datasets is to weigh the classes, where lower weights are given to the most frequent ones. Instead, for the least frequent class, higher weights result in a greater loss function output in case of misclassification. This can be done using a weighted loss function. Another approach, much more naive, is to drop training samples that

belong to the most frequent class. In a domain like ECG beat classification, where there is such huge unbalancing, the choice between the two approaches cited above is not trivial. We can observe that for a single patient, the peaks that belong to the normal-peak class are very similar to each other, at the human eye they may seem indistinguishable. This is very different from i.e. an image classification task where we have unbalancing between the images of cats and images of dogs, because each image belonging to the same class clearly differs from each other. Given the above and because of empirical results, we opted for a "Drop and resample" approach. We dropped samples of class N (we dropped a portion of N peaks of each patient) and then we resampled examples of S and V with duplication to obtain a perfectly balanced dataset. eq. (2) is used to calculate the number of N samples to drop, while eq. (3) is used to get the number of samples to duplicate for classes S and V. Because each patch has an uniform probability to be chosen, there may be samples that are not duplicated and some that are duplicated more than once.

$$|N|_{drop} = |N| - max\{|S|, |V|\} \times 2 \tag{2}$$

This led us to have 14985 patches for each class.

$$\begin{aligned} |S|_{resample} &= |N| - |N|_{drop} - |S| \\ |V|_{resample} &= |N| - |N|_{drop} - |V| \end{aligned} \tag{3}$$

3.5 Models

Different models, from classical ML and three neural networks, are implemented and their performances compared in order to find the one that most generally performed the right classification. Here are described the implementations of the three neural networks that are, in practice, a one-dimensional ResNet, similar to the one implemented by Kachuee et al. [5], an Autoencoder used to perform dimensionality reduction (fig. 14) and a CNN with LSTM inspired by the model of Sigurthorsdottir et al. [14].
The ResNet is implemented as follows: the 350 samples of each patch are fed as input and then convolved using 64 filters with a size of 15. Then batch normalization is applied before the ReLU activation function and global max pooling. Then two ResNet identity blocks, which implementation is illustrated in fig. 13, are placed. The number of filters used for convolution in these blocks is fixed to 64, with a kernel of size 7. After this, three sequences of ResNet convolution block plus ResNet identity block are implemented by doubling the number of filters each step. At the bottom of the network, the classification is performed through an FC layer with a softmax activation function. The model is trained using a batch size of 32 and Adam optimizer with a learning rate of 10^{-5}. The low learning rate is due to the fact that with a higher value the training was not representative, oscillating between pseudo-optimal points, as can be seen in appendix A.2 (fig. 21, fig. 22). The categorical cross-entropy loss function is used, also to monitor the training through early stopping.

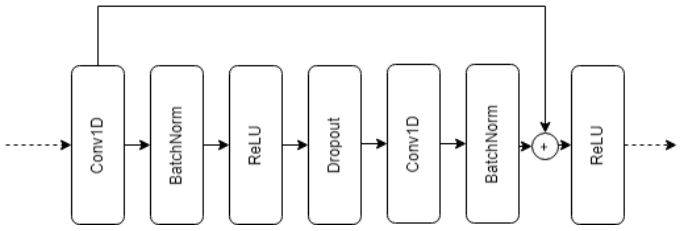

Figure 13: Architecture of the ResNet identity block.

The AutoEncoder architecture (fig. 14), instead, consist of an FC network that first encodes the input (fig. 15) with a compression factor of 8.75, and is trained to decode the encoded features, with the aim of reconstructing (fig. 17) the input patch. The encoder is composed of a sequence of Dense layers with, respectively, 512, 256, and 40 neurons, with a dropout layer between the first and second

hidden layers. The last layer neurons identify the compressed dimension to which the input is reduced. The decoder section is the mirrored architecture of the encoder, that learns how to reconstruct the input given the compressed features (fig. 16). After the autoencoder training, the decoder section is detached and substituted by an appropriate model in order to perform the classification, using as input the output of the encoder. Different models from classic ML (SVC, KNN, RFC) among with a simply FFNN are compared for this purpose. Obviously, each classifier needs to be trained using the compressed patches as input and being supervised with the annotations of each peak. The autoencoder is trained using a batch size of 256 and Adam optimizer with a learning rate of 10^{-4} and mean square error loss function is used.

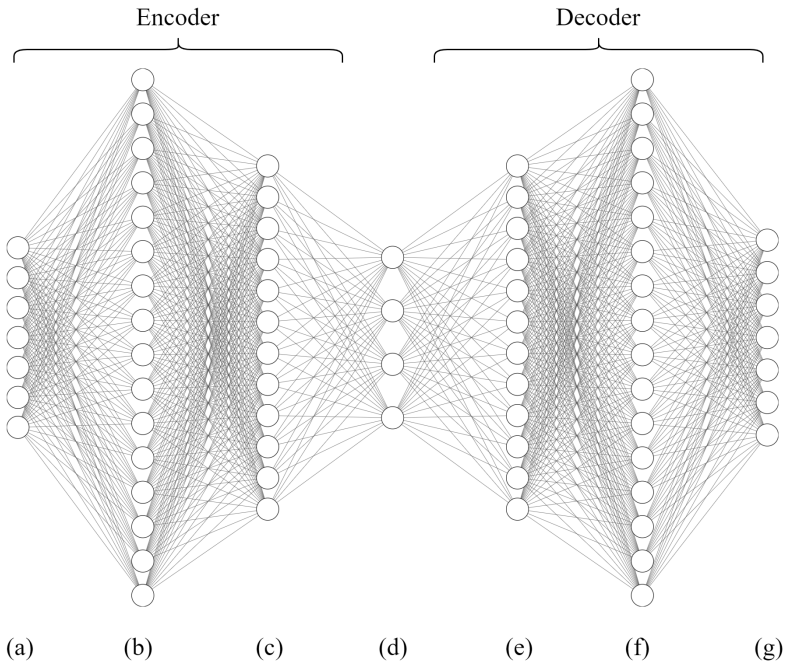

Figure 14: Architecture of the autoencoder used to encode the input patch. The structure is described as follows: (a) Input layer of 350 units. (b)-(f) hidden layers of 512 units. (c)-(e) hidden layers of 256 units. (d) layer that represents the encoding of the input over 40 units. (g) decoded layer, same dimension of the input one. A dropout with rate 10% is applied to layers (b) and (e).

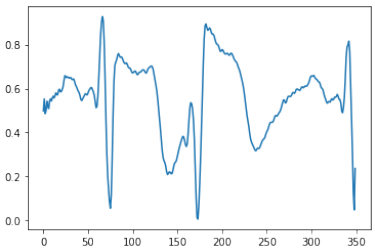

Figure 15: Patch extracted that is fed as input to the autoencoder.

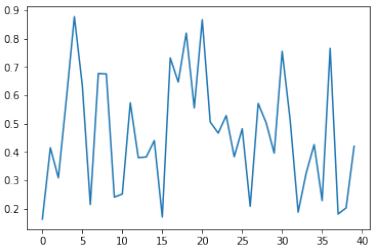

Figure 16: Visualization of encoding performed on (fig. 15) sample.

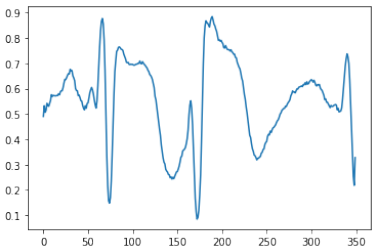

Figure 17: The same patch of (fig. 15) after being processed by the trained autoencoder.

For the CNN with LSTM instead, being inspired by Limam et al. [7], a sort of One vs One approach is adopted, making a cascade classification. This is done to highlight differences between

normal and abnormal beats. First of all samples of PVC and PAC are merged in a unique class of abnormal peaks and the network [14] is trained to predict if samples are normal or abnormal. Then all the patches of normal peaks are discarded and the same network is trained to predict if the samples are PVC or PAC. To make the prediction first it is observed how the first network predict the samples and then, if a sample are predicted as abnormal, a prediction with the second network is made. In particular, as can be seen in fig. 18, the network used for both the models is composed of three blocks of convolution respectively with 32, 64, and 128 filters and 25, 12, 9 kernel size, then there is a max-pooling with pool size 2 for every block and at the end a dropout layer with rates 0.05, 0.1 and 0.15. At the end of the three blocks, there are two LSTM with 64 units and a dense layer with 1 unit and a sigmoid activation function. Both the models are trained with batch size 128 and Adam optimizer with a learning rate 10^{-5}. As loss function, the binary cross-entropy is used and also in this case the training is monitored through early stopping. It is important to highlight that this approach is possible only under the conditions of this specific task, being only three classes containing exactly two abnormalities. The approach quickly becomes infeasible when more classes are introduced, if using classifiers of such complexity.

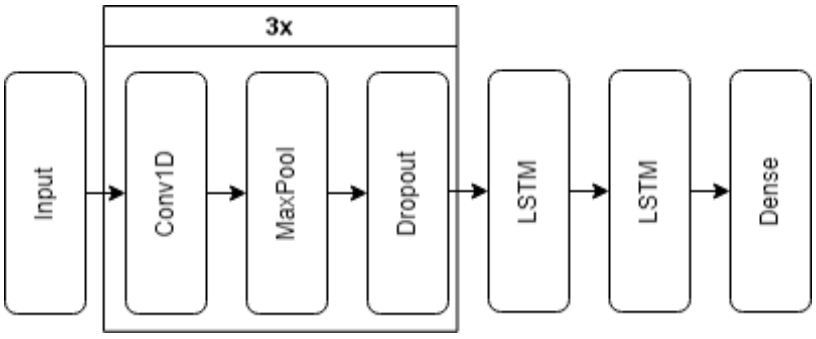

Figure 18: Architecture of each LSTM CNN used in cascade approach.

3.6 Model assessment

To assess the model performances we split the entire dataset, after the removal of outlier patients, in training, validation and test set, with a ratio of 80-10-10. This results in 79 patients for training, 10 for the validation set and 10 to test performances of the best model (6 patients are identified as outliers). An important note is that the split is performed patient-level and not after extracting the patches. This is necessary because the same patient beats are very similar to each other and that would result in a validation set that is very similar to the training samples that the model has been trained on, resulting in biased and misleading performance metrics. After performing the split, patches are extracted and the "drop and resample" method, described in section 3.4, is applied to the patches belonging to the training set. Metrics used to assess the performances are: precision, recall, F1-score and accuracy (everything on class level being the validation and test sets unbalanced). Formulas for those metrics are reported in appendix A. The best model is selected using performances on the validation set, then a final prediction on the test set (of the selected model) is done to report performances.

The complete pipeline that summarizes everything mentioned above is reported in appendix A.2 (fig. 20)

4 Results

Different models have been tried as mentioned in section 3.5, measuring the prediction performance over the same validation set. All models, except LSTM-CNN and ResNet, take the first lead patches

Model	Parameter	Value
SVC	C	1
	kernel	radial basis
	decision function	one vs rest
KNN	neighbors	3
RFC	weight	distance
	estimators	200
NN	hidden1 units	512
	hidden2 units	512
	dropout	10%
	batch size	256
	learning rate	1e-4
	decay rate	0.75
	decay steps	25 epochs
AutoEncoder	see section 3.5	
ResNet1D	see section 3.5	
LSTM-CNN	see section 3.5	

Table 1: Model parameters used in running experiments reported in table 2.

in input as the usage of both signals didn't improve the performances. The opposite behaviour is encountered with the above-mentioned neural networks. Results are reported in table 2. We can see that the best models leverage the encoded representation of the first lead patches given by the trained autoencoder (fig. 14). ResNet's and CNN's parameters are described in section 3.5 while other parameters are listed in table 1.

Results on the validation set show that the usage of the AutoEncoder representation improves performances if we consider the same classifier. Models like the random forest classifier and ResNet have better precision results, but this is due to the fact that they are good at avoiding misclassification for the N class. The validation set, contrary to the training set, is still unbalanced, so it's normal to have such low values of precision even for models that have good recall metrics such as the SVC. ResNet is the model that has more "best-values", but for the S class does not perform well and that could be seen in both recall and accuracy metrics. The SVC that leverages the autoencoder representation is the model that overall maintains good performance on all the metrics, thus it will be the model of our choice. Performances obtained on the test set are summarized in table 3 and the confusion matrix is reported in fig. 19.

5 Discussion

It's hard to compare obtained results with state-of-the-art performances because of the quality of the given dataset. The standard dataset used to measure arrhythmias classification performances is the MIT-BIH database, which incorporates 47 patients' annotated ECGs. Cardiologist-level performances obtained by Pranav Rajpurkar et. al [11] could be related to the availability of 30000 patients' annotated ECGs. In contrast, they were able to identify much more types of arrhythmias (13 excluded the normal sinus rhythm), while we are limited to two types (PAC and PVC). We also think that many studies in the literature show astonishing performances (above 99% of accuracy) because they perform the validation split after extracting the heartbeat instances. That is an error and leads to misleading metrics because for a single patient, heartbeats are very similar to each other, consequently, the validation set will contain samples that are too alike to examples used in training. The same considerations have been made by Eduardo José da S. Luz et al. in an exhaustive survey about ECG classification [12]. Given the above, we consider our performances a good result, but there are margins of improvement. One of our main problems is due to the domain and is the similarity between N and PAC peaks. This similarity has been mitigated using a patch extraction

Model	Class	Precision	Recall	F1-Score	Accuracy	Support
SVC	N	**1.00**	0.93	0.96	0.93	20924
	S	0.46	0.85	0.60	0.85	1032
	V	0.69	0.93	0.79	0.93	1703
KNN	N	0.99	0.92	0.95	0.92	20924
	S	0.44	0.77	0.56	0.77	1032
	V	0.55	0.82	0.66	0.82	1703
RFC	N	**1.00**	0.86	0.92	0.97	20924
	S	0.65	0.70	0.68	0.7	1032
	V	**0.76**	0.84	**0.80**	0.84	1703
AE + SVC	N	0.99	0.94	0.97	0.94	20924
	S	0.56	**0.86**	0.68	**0.86**	1032
	V	0.70	0.92	0.79	0.92	1703
AE + KNN	N	0.99	0.92	0.96	0.92	20924
	S	0.49	0.79	0.60	0.79	1032
	V	0.57	0.84	0.68	0.84	1703
AE + RFC	N	**1.00**	0.84	0.91	**0.98**	20924
	S	**0.84**	0.66	0.74	0.66	1032
	V	0.73	0.74	0.74	0.74	1703
AE + NN	N	0.99	0.93	0.96	0.93	20924
	S	0.54	0.85	0.66	0.85	1032
	V	0.61	0.89	0.73	0.89	1703
ResNet 1D	N	**1.00**	**0.96**	**0.98**	0.96	20924
	S	0.83	0.29	**0.75**	0.69	1032
	V	0.62	**0.98**	0.76	**0.98**	1703
LSTM-CNN	N	0.98	**0.96**	0.97	0.96	20924
	S	0.53	0.62	0.57	0.62	1032
	V	0.76	0.82	0.79	0.82	1703

Table 2: Result after performing the prediction over the validation set for the following models: *SVC* (support vector classifier), *KNN* (K nearest neighbors classifier), *RFC* (random forest classifier), *AE* (autoencoder to encode the input), *NN* (feedforwrd neural network), *ResNet1D* (residual deep CNN described above). Values in **bold** are best for each class and metric. Formulas for reported metrics can be found in appendix A. *Support* is the number of samples tested.

Model	Class	Precision	Recall	F1-score	Accuracy	Support
AE+SVC	N	1.00	0.96	0.98	0.96	23231
	S	0.43	0.91	0.59	0.91	619
	V	0.82	0.91	0.86	0.91	1016

Table 3: Results obtained on the test set.

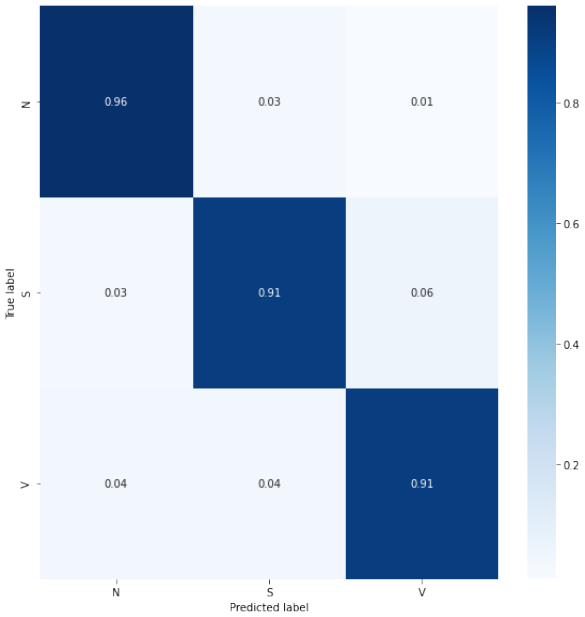

Figure 19: Confusion matrix for **AE+SVC** over the test set.

that leverages the average period between peaks, but this is not a silver bullet as misclassification persists.

While dealing with ResNet training, the relatively small number of patients compared with the higher number of single peaks had impacted the learning process. As discussed in section 3.5, with a higher learning rate the process does not proceed constructively but led to relatively acceptable results. This is not reported because it is not reliable. As the confusion matrix shows, reported in appendix A.2 (fig. 23), the network does not generalize enough the task. This problem is also highlighted by the very low value of recall (see table 2), which leads to strong overfitting on a specific class. This may arise from the complexity of ResNet architecture, which strongly depends on the quality of input and, particularly, of the data that represent each class. The fact that PAC beats are strongly mispredicted is in accordance with the domain considerations. The problem does not affect the autoencoder architecture that learns how to extract relevant features in the encoding phase. The full model relies on them as input to perform the classification task, giving a certain level of tolerance on the original patch and thus augmenting the model generalization power.

6 Conclusion

This work describes with a top-down analysis each phase that is encountered dealing with AI models and their medical applications. It starts with an aware analysis of data to comprehend what the experiment must deal with. This kind of knowledge had moved the preprocessing of the dataset, from filtering to input patches extraction, passing through outliers removal and dataset balancing. The solutions proposed are implemented to face the characteristics of the available samples and aim to develop a general and robust model. The experiment shows the comparison of some different architecture and assesses how these models performed the task. With this analysis were possible to present the model that, as expected, maintained comparable performances also on the test set.
The growth of AI had made it possible for its application in a lot of different fields. The usage of such technologies in such intimate and critical field as medicine need to pose attention to topics that are sometimes poorly considered. Discussions about privacy, dataset biases, and transparency of the models acquire much more importance than ever. The latter, for instance, is desirable to let clinicians understand in a comprehensible manner the decision taken by the AI system, to preserve

patient safety, care and to inform him/her comprehensively. On the other hand, the medical field itself is pushing toward a more digitized and accessible kind of service. This change aims to enhance patient consideration while leading the way to the use of AI, thanks to the huge amount of data that potentially can be retrieved. This experiment highlights the strict dependency between the quality of data and the performance of the system. More attention on the domain is needed to apply a wise preprocessing and data analysis. The variety of the same medical data from different patients, in general, suits well with neural networks' capability of generalization in extracting relevant features out of inputs to perform a specific task. Related to this, another point arises from this work. The dataset was characterized by a huge number of peaks coming from a relatively small number of patients. To increase the model generalization power, it is preferable to access as many subjects as possible. Since this is not always a viable option, it is needed to find a compromise. The evolution of digital health care goes along with digital evolution, and the same consideration can be easily made about AI systems and their medical applications. The potentialities of such models on such different tasks are widely shown in research, but the time of their effective application in medical routines in not yet arrived.

6.1 Implementation acknowledgements

The experiment is implemented in Python with the help of some useful libraries. In fact *pandas* [15] is used to easily manage the dataset through its dataframe objects, *numpy* [3] to handle the datasamples, *hearthpy* [2] to scale the signals during preprocessing, *scikit-learn* [10] to implement classical ML models and to handle the train/test/validation split and, finally, *tensorflow* and *keras* [8] to implement the neural networks and handle the training and inference through them.

References

[1] Syed Anwar **andothers**. "Arrhythmia Classification of ECG Signals Using Hybrid Features". **in***Computational and Mathematical Methods in Medicine*: 2018 (**november** 2018), **pages** 1–8. DOI: 10.1155/2018/1380348.

[2] Paul van Gent **andothers**. "Analysing Noisy Driver Physiology Real-Time Using Off-the-Shelf Sensors: Heart Rate Analysis Software from the Taking the Fast Lane Project." **innovember** 2018: DOI: 10.13140/RG.2.2.24895.56485.

[3] Charles R. Harris **andothers**. "Array programming with NumPy". **in***Nature*: 585.7825 (**september** 2020), **pages** 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[4] Nathalie Japkowicz. "The class imbalance problem: Significance and strategies". **in***Proc. of the Int'l Conf. on Artificial Intelligence*: **volume** 56. Citeseer. 2000.

[5] Mohammad Kachuee, Shayan Fazeli **and** Majid Sarrafzadeh. "ECG Heartbeat Classification: A Deep Transferable Representation". **inapril** 2018.

[6] Rahul Kher. "Signal Processing Techniques for Removing Noise from ECG Signals". **in**2019.

[7] Mohamed Limam **and** Frederic Precioso. "Atrial fibrillation detection and ECG classification based on convolutional recurrent neural network". **in***2017 Computing in Cardiology (CinC)*: 2017, **pages** 1–4. DOI: 10.22489/CinC.2017.171-325.

[8] Martín Abadi **andothers**. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[9] V.K. Murthy **andothers**. "Clinical Usefulness Of ECG Frequency Spectrum Analysis". **in***The Second Annual Symposium on Computer Application in Medical Care, 1978. Proceedings.* 1978, **pages** 610–612. DOI: 10.1109/SCAMC.1978.679974.

[10] F. Pedregosa **andothers**. "Scikit-learn: Machine Learning in Python". **in***Journal of Machine Learning Research*: 12 (2011), **pages** 2825–2830.

[11] Pranav Rajpurkar **andothers**. *Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks*. 2017. arXiv: 1707.01836 [cs.CV].

[12] Eduardo José da S. Luz **andothers**. "ECG-based heartbeat classification for arrhythmia detection: A survey". **in***Computer Methods and Programs in Biomedicine*: 127 (2016), **pages** 144–164. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2015.12.008. URL: https://www.sciencedirect.com/science/article/pii/S0169260715003314.

[13] Çağla Sarvan **and** Nalan Özkurt. "ECG Beat Arrhythmia Classification by using 1-D CNN in case of Class Imbalance". **in***2019 Medical Technologies Congress (TIPTEKNO)*: 2019, **pages** 1–4. DOI: 10.1109/TIPTEKNO.2019.8895014.

[14] Halla Sigurthorsdottir **andothers**. *ECG Classification with a Convolutional Recurrent Neural Network*. 2020. arXiv: 2009.13320 [cs.LG].

[15] The pandas development team. *pandas-dev/pandas: Pandas*. **version** 1.1.5. **february** 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

A Appendix

A.1 Formulas

Considering:

- $TP(C_k)$ (true positives for class C_k)
- $FP(C_k)$ (false positives for class C_k)
- $TN(C_k)$ (true negatives for class C_k)
- $FN(C_k)$ (false negatives for class C_k)

Performance metrics:

- $Accuracy(C_k) := \frac{TP(C_k)+TN(C_k)}{TP(C_k)+FP(C_k)+FN(C_k)+TN(C_k)}$
- $Precision(C_k) := \frac{TP(C_k)}{TP(C_k)+FP(C_k)}$
- $Recall(C_k) := \frac{TP(C_k)}{TP(C_k)+FN(C_k)}$
- $F1-score(C_k) := \frac{TP(C_k)}{TP(C_k)+\frac{FN(C_k)+FP(C_k)}{2}}$

A.2 Figures

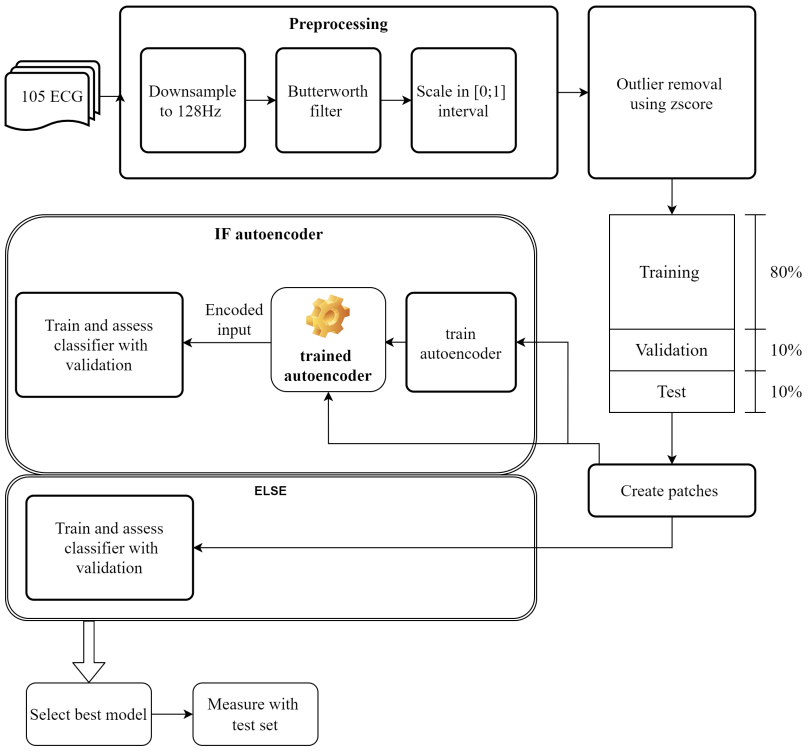

Figure 20: Complete pipeline of the task.

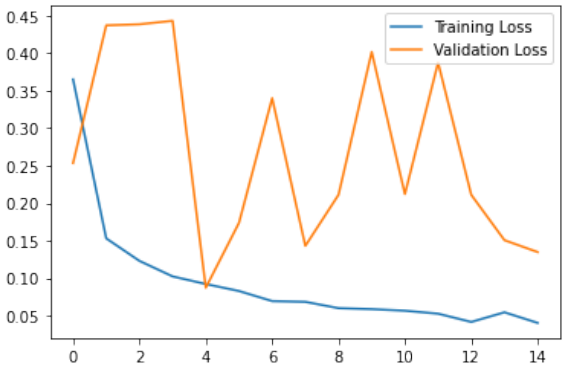

Figure 21: Plot of ResNet training with a learning rate of 10^{-3}.

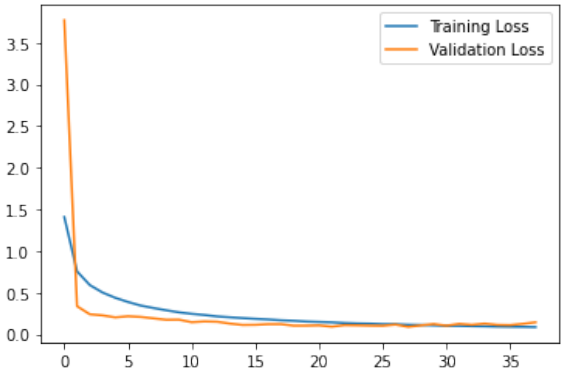

Figure 22: Plot of ResNet training with a learning rate of 10^{-5}.

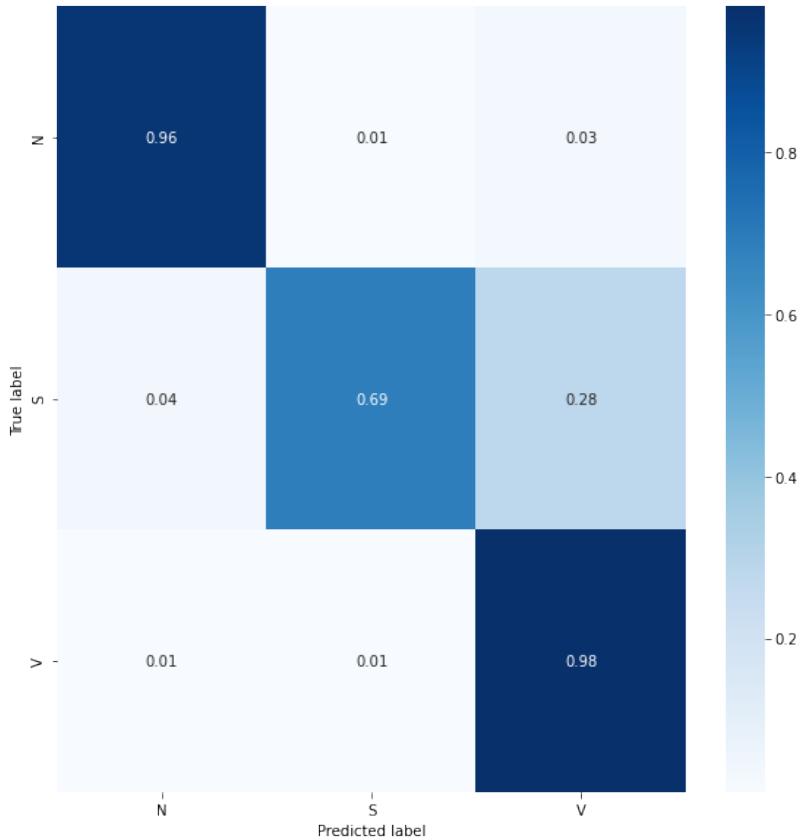

Figure 23: ResNet confusion matrix computed on unbalanced validation set.