Atrial Fibrillation Detection and ECG Classification based on Convolutional Recurrent Neural Network

Mohamed Limam, Frederic Precioso

Université Côte d'Azur, CNRS, I3S, France

Abstract

The aim of the 2017 PhysioNet/CinC Challenge [1] is to classify short ECG signals (between 30 seconds and 60 seconds length), as Normal sinus rhythm (N), Atrial Fibrillation (AF), an alternative rhythm (O), or as too noisy to be classified.

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) as classifiers have recently shown improved performances compared to methods established in various sound recognition tasks [2] and interesting result in tasks such as the 2016 Physionet Challenge for the classification of heart sound [3].

Our approach is based on a convolutional recurrent neural network (CRNN), involving two independent CNNs, to extract relevant patterns, one from the ECG and the other from the heart rate, which are then merged into a RNN accounting for the sequence of the extracted patterns. The final decision is then evaluated through a Support Vector Machine (SVM).

1. Introduction

The main objective of the 2017 PhysioNet/CinC is to classify cardiac signals in four different classes: N (*Normal sinus rhythm*), AF (*Atrial Fibrillation*), O (an alternative rhythm), or "too noisy" (to be recognized). The data provided by the challenge are electrocardiograms (ECG). An electrocardiogram shows the heart's electrical activity. The spikes and dips are called waves. It allows to highlight various cardiac abnormalities and has an important place in diagnostic tests in cardiology. Electrocardiogram immediately provides heart health information. When reading an electrocardiogram, five characteristic waves can be observed: P, Q, R, S and T. The P wave represents atrial depolarization, the QRS complex corresponds to the depolarization and contraction of the ventricles, right and left and the T wave corresponds to the repolarization phase of the two ventricles [6, 7, 8, 9].

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) as classifiers have recently shown improved performances compared to methods established in various sound recognition tasks [2] and interesting result in tasks such as the 2016 Physionet Challenge for the classification of heart sound [3].

2. Data preprocessing

To train our model we have 8528 ECGs at our disposal. Thanks to the Matlab code provided by the challenge [4], we have generated features useful for the processing of our data such as the position of the R peaks in the signal and the heart rate for each signal.

Each signal has been divided into several windows containing the most important elements of an electrocardiogram, which are P, Q, R, S and T waves. To create these windows, we have considered as starting (green points in the Figure 1) and ending (black points in the Figure 1) points of the windows, the middle of each R-R interval in the signal. The R-R interval is what connects two consecutive R waves (red points in Figure 1). If the distance between two R peaks is too large, a threshold has been set at about 0.5 milliseconds before (fore the starting point) and after (for the ending point) each R peak.

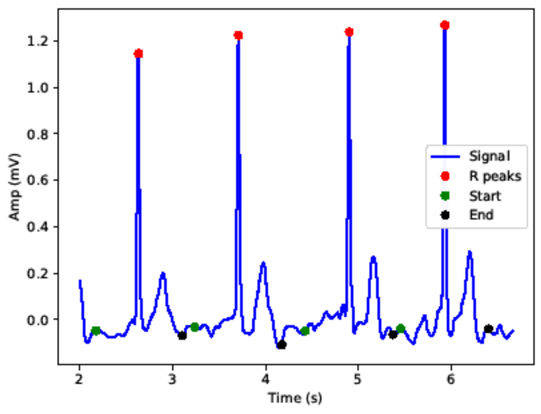

Figure 1. A portion of an AF signal with the R peaks, the starting points of each window in green and the ending points in black.

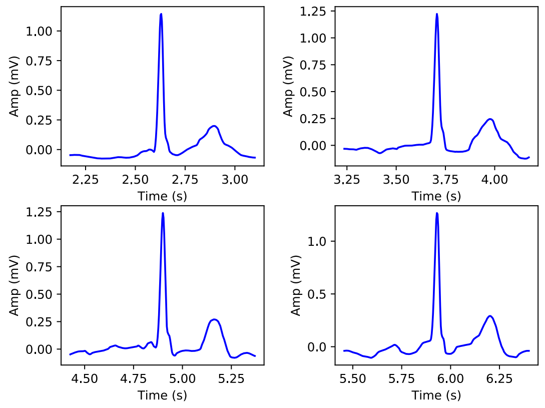

Figure 2. Windows extract from the AF signal.

We rescale the heart rate between [0, 1].

3. Model

For reducing the problem of multiclass classification we have decided to implement a strategy *One-Vs-All.* We consider the problem as a multiple binary classification problem.

We have implemented a hierarchical model consisting of three CRNNs. The first model classifies the AF class against the rest of the classes (i.e. N, O and Noisy). Then we discard the AF samples from the dataset and we train a second model which classifies the N class against the rest. Here the remaining samples are from O and Noisy classes. We repeat a third time for the last model we discard the samples of the class N and we train the last model which classifies the O class against the Noisy class.

To know the class of a signal it will be necessary to start by making a prediction on the first model. If the model predicts that the signal belongs to the class AF we stop here. Otherwise we return the signal to the second model. If the model predicts that the signal belongs to the N class, we stop at this level. Or it will be necessary to make a last prediction on the last model to know if the signal belongs to the O class or to the Noisy class.

3.1 Architecture

We describe here the structure of the CRNN. The model has two inputs: one for the ECG and the other for the heart rate. Each input is then represented by high-level features given by the output of a CNN. These high-level features are extracted separately, then concatenated and input into a RNN. Feature extraction is performed by 1-dimensional convolution layers leading to a 1-dimensional high-level feature vector output which is then seen as a sequence of feature values and sent to a recurrent layer to learn the longer-term temporal dependencies between relevant patterns present in the features.

Layers	Parameters	Activ.
Convolution 1D	32	ReLU
Max-Pooling 1D	2	
Dropout	0.05	
Convolution 1D	64	ReLU
Max-Pooling	2	
Dropout	0.1	
Convolution 1D	128	ReLU
Max-Pooling 1D	2	
Dropout	0.15	
Merge / Concatenation		
Masking Layer		
LSTM	64	Tanh
LSTM	64	Tanh
Dense Layer	2	Softmax

Table 1. Convolutional Recurrent Neural Network layers and parameters.

The ECG and the heart rate data have sequences of different lengths. Before being input to our architecture, the sequences are zero padded. This leads to use a Masking Layer in the network in order to discard dimensions of the output feature vector resulting from zero padded processed values that we do not want the recurrent network to take into account as sequence values.

First the input is sent to three 1D convolutional layers. Between each convolutional layer we add max-pooling layer which extracts the maximum value of the filters and thus provide the most informative features while avoiding redundancy and reducing computational cost thanks to data reduction. In addition to max-pooling layers, dropout have been added to reduce overfitting. For the convolution we used kernels of size 3.

For the next step, we use RNN, more precisely Long Short-Term Memory (LSTM) [5]. Unlike a conventional RNN the LSTM solve two problems: LSTM can handle long-term dependencies and solve vanishing gradient issue. We have 2 consecutive LSTM layers in order to increase the length of time dependencies.

The last layers here are a dense layer with two outputs followed by a softmax layer. The two outputs correspond to the binary classification of the *One-Vs-All* strategy for each class.

All these layers are detailed in the Table 1 above and a diagram presents their organization in Figure 3 below.

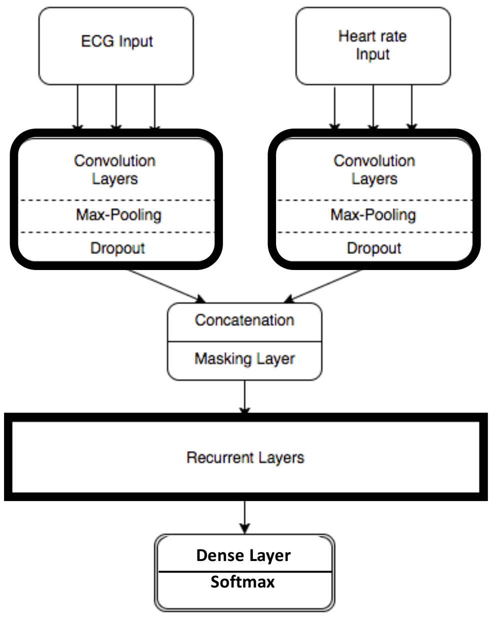

Figure 3. Our CRNN Model. The bold boxes are repeated processes.

3.2. Training

For training our models we split our data into training and validation set. We use 85 % for training and 15 % for validation. The validation is composed of 15% of each class.

Convolutional and LSTM layers are using Glorot uniform weight initialization [10]. The dense layer is using He normal weight initialisation [11]. During training we used cross entropy as loss function and *Adam* as optimizer [12] with a fixed learning rate of 10^{-4}. Our architecture has been implemented using *Keras* and *Tensorflow* as backend.

3.3. Final classification

Deep learning is only a data representation methodology on which decision algorithms can be applied. Here we have decided to use a SVM, which is known to be a powerful binary classifier. We have recovered the features generated by the last layer of LSTM and we have trained the SVM on these new representations.

4. Results

Here are the results of the three models for the different classes, for the CRNN and CRNN-SVM. We use 85% of the data for training and 15% for validation.

Table 2. AF vs All model train result.

AF vs All	Sensitivity	Specificity	Score
CRNN	**0.825**	0.987	**0.906**
CRNN-SVM	0.817	**0.988**	0.902

Table 3. N vs All model train result.

N vs All	Sensitivity	Specificity	Score
CRNN	0.928	**0.917**	**0.922**
CRNN-SVM	**0.947**	0.886	0.916

Table 4. O vs Noisy model train result.

O vs Noisy	Sensitivity	Specificity	Score
CRNN	0.979	**0.796**	**0.887**
CRNN-SVM	**0.994**	0.771	0.882

Table 5. AF vs All model validation result.

AF vs All	Sensitivity	Specificity	Score
CRNN	**0.727**	0.986	**0.856**
CRNN-SVM	0.723	**0.987**	0.855

Table 6. N vs All model validation result.

N vs All	Sensitivity	Specificity	Score
CRNN	0.879	**0.847**	**0.863**
CRNN-SVM	**0.907**	0.801	0.854

Table 7. Other vs Noisy model validation result.

O vs Noisy	Sensitivity	Specificity	Score
CRNN	0.969	0.666	0.817
CRNN-SVM	**0.983**	**0.672**	**0.827**

The non-definitive result obtained with the CRNN on the validation dataset is 0.77. With the CRNN-SVM the result obtained is also 0.77. However, if we obtain the same results respectively 0.78 for AF and 0.89 for the Normal classes, the results on the Other class vary. With the CRNN we get 0.63 while with the CRNN-SVM the score for the Other class is 0.65.

5. Discussion

We have presented our deep hierarchical model for the Physionet 2017 Challenge. The performance differs somewhat between our results during the training of our model and the validation on unseen data. Notably for the class Other where we noticed that it is mistaken for the class AF and even more for the class Normal. Using the SVM for the final decision allows us to improve the results of the third model trained on the Other and Noisy classes. For the first two models, the results remain similar with a softmax layer or with an additional "SVM Layer". The network struggles to generalize the Other class maybe because it gathers many different heart diseases, so the variability intra-class is too impacting. The Normal class, being the most represented among the samples, offers the best results. The final score on the hidden challenge dataset is 0.77, like previously on the subset. The question arises then if, with a larger dataset, the results would be better? This would imply a wider neural network. In addition, the processing of mirrored electrocardiograms has not been done correctly, this would improve the results.

Acknowledgements

The authors are grateful to Inria Sophia Antipolis – Méditerranée "Nef" computation cluster for providing resources and support.

References

[1] Clifford, G., Liu, C., Moody, B., Silva, I., Li, Q., Johnson, A. & Mark, R. (2017). AF Classification from a Short Single Lead ECG Recording: the PhysioNet Computing in Cardiology Challenge 2017. Computing in Cardiology (Rennes: IEEE), Vol 44, 2017 (In Press).

[2] Parascandolo, G., Heittola, T., Huttunen, H., & Virtanen, T. (2017). Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25(6), 1291-1303.

[3] Thomae, C., & Dominik, A. (2016). Using deep gated RNN with a convolutional front end for end-to-end classification of heart sound. In Computing in Cardiology Conference (CinC), 2016 (pp. 625-628). IEEE.

[4] Sarkar, S., Ritscher, D. & Mehra, R. (2008). A detector for a chronic implantable atrial tachyarrhythmia monitor, IEEE Trans Biomed Eng 55 (3) (2008) 1219–1224.

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[6] Rekik, S., & Ellouze, N. (2015). P-Wave Detection Combining Entropic Criterion and Wavelet Transform. Journal of Signal and Information Processing, 6(3), 217.

[7] Guyton, A.C. & Hall, J.E. (2006). Textbook of Medical Physiology, 11 ed Elsevier Saunders, 2006.

[8] Kiranyaz, S., Ince, T., Hamila, R., & Gabbouj, M. (2015). Convolutional neural networks for patient-specific ECG classification. In Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE (pp. 2608-2611). IEEE.

[9] Wu, H., & Prasad, S. (2017). Convolutional Recurrent Neural Networks for Hyperspectral Data Classification. Remote Sensing, 9(3), 298.

[10] Glorot, X., Bordes, A. & Bengio, Y. (2011). Deep sparse rectifier neural networks. International Conference on Artificial Intelligence and Statistics. 2011.

[11] He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.

[12] Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR 2015.

Address for correspondence:

Mohamed Limam
Frederic Precioso
Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) - UMR7271 - UNS CNRS, 2000, route des Lucioles - Les Algorithmes - bât. Euclide B 06900 Sophia Antipolis - France