elektron ámbar

Revista elektron, Vol. 4, No. 2, pp. 52-57 (2020)
https://doi.org/10.37537/rev.elektron.4.2.101.2020

Original Article

Deep Recurrent Learning for Heart Sounds Segmentation based on Instantaneous Frequency Features

Aprendizaje profundo y recurrente para la segmentación de sonidos cardíacos basado en características de frecuencia instantánea

Álvaro Joaquín Gaona*[1], Pedro David Arini*†[2]

*Facultad de Ingeniería, Universidad de Buenos Aires,
Instituto de Ingeniería Biomédica, (IIBM)
Avenida Paseo Colón 850, C1063ACV, Buenos Aires, Argentina
[1]agaona@fi.uba.ar

† Instituto Argentino de Matemática "Alberto P. Calderón", CONICET
Saavedra 15, C1083ACA, Buenos Aires, Argentina
[2]pedro.arini@conicet.gov.ar

Recibido: 15/08/20; Aceptado: 31/10/20

***Abstract*—In this work, a novel stack of well-known technologies is presented to determine an automatic method to segment the heart sounds in a phonocardiogram (PCG). We will show a deep recurrent neural network (DRNN) capable of segmenting a PCG into their main components and a very specific way of extracting instantaneous frequency that will play an important role in the training and testing of the proposed model. More specifically, it involves an Long Short-Term Memory (LSTM) neural network accompanied by the Fourier Synchrosqueezed Transform (FSST) used to extract instantaneous time-frequency features from a PCG. The present approach was tested on heart sound signals longer than 5 seconds and shorter than 35 seconds from freely-available databases. This approach proved that, with a relatively small architecture, a small set of data and the right features, this method achieved an almost state-of-the-art performance, showing an average sensitivity of 89.5%, an average positive predictive value of 89.3% and an average accuracy of 91.3%.**

Keywords: phonocardiogram; fourier synchrosqueezed transform; long short-term memory.

***Resumen*—En este trabajo se presenta un conjunto de técnicas bien conocidas definiendo un método automático para determinar los sonidos fundamentales en un fonocardiograma (PCG). Mostraremos una red neuronal recurrente capaz de segmentar segmentar un fonocardiograma en sus principales componentes, y una forma muy específica de extraer frecuencias instantáneas que jugarán un importante rol en el entrenamiento y validación del modelo propuesto. Más específicamente, el método propuesto involucra una red neuronal *Long Short-Term Memory* (LSTM) acompañada de la Transformada Sincronizada de Fourier (FSST) usada para extraer atributos en tiempo-frecuencia en un PCG. El presente enfoque fue evaluado con señales de fonocardiogramas mayores a 5 segundos y menores a 35 segundos de duración extraídos de bases de datos públicas. Se demostró, que con una arquitectura relativamente pequeña, un conjunto de datos acotado y una buena elección de las características, este método alcanza una eficacia cercana a la del estado del arte, con una sensitividad promedio de 89.5%, una precisión promedio de 89.3% y una exactitud promedio de 91.3%.**

Palabras clave: fonocardiograma; transformada sincronizada de fourier; *long short-term memory*.

I. Introduction

Phonocardiography is a method to record the acoustic phenomena of the heart graphically. It is used to provide information about the cardiac cycle by plotting sounds and murmurs of the heart. The sounds result from the closure of the heart valves, and it is possible to identify at least two sounds. The first one, S_1, corresponds to the closure of the atrioventricular valves (mitral and tricuspid valve) at the beginning of the systole. At this point, the ventricles filled with blood from the atriums and muscle contractions begin to eject the oxygenated and deoxygenated blood to the pulmonary and systemic circuits respectively. After most of the blood has been ejected from the ventricles, the aortic and pulmonary valves close producing the second sound, S_2. Additionally, two other segments of the phonocardiogram (PCG) can be identified. The first one is the segment S_1-S_2 called isovolumetric contraction and the second one is the segment. S_2-S_1 called isovolumetric relaxation, which usually is shorter than the first segment. Heart sounds segmentation dates back to 1997 where H. Liang *et al.* used a deterministic algorithm based on the normalized average Shannon energy of a PCG signal achieving a 93% correct ratio. This approach has, however, some drawbacks such as corrupting noise. In the same year, H. Liang *et al.* [1] proposed an algorithm based on wavelet decomposition and reconstruction performing correctly in over 93% of cases. Heart sounds segmentation boomed in 2010 when Schmidt *et al.* [2] proposed a Hidden Markov Model (HMM) based on time-duration called Dependent-duration Hidden Markov Model (DHMM). Additionally, it introduced the use of annotations derived from the EKG to label training sets to train the proposed model, later used by Springer *et al.* [3] to go even further and outperform the previous work by adding logistic regression and modifying the implementation of the Viterbi algorithm. In 2018, Renna *et al.* in [4] have used Deep learning techniques to segment the PCG. Their

approach, motivated by a novel convolutional neural network called U-net [5] for neuronal structures segmentation in electron microscopic stacks, used Schmidt and Springer techniques such as labelling and feature extraction (Homomorphic envelogram, Hilbert envelope, Wavelet envelope and Power Spectral Density envelope) to outperform what was at that time known as a state-of-the-art technique.

In this work, we propose an implementation based on a DRNN to segment PCG signals into their main components. The proposed method involves a Long Short-Term Memory (LSTM) neural network accompanied by the Fourier Synchrosqueezed Transform (FSST) used to extract instantaneous time-frequency features in a PCG.

II. Material and Methods

Most of the work begins at deciding which approach and what data is going to be used to achieve the desired goal.

Deep Learning has proved to the world that it is a compelling strategy to address various kind of problems and signal segmentation is no exception. Moreover, deep learning techniques have not been used in PCG as it has been used in electrocardiogram (EKG) signals lately. Long Short Term Memory neurons are pretty powerful neural network units that can achieve great accuracy when trained on time-series data. This data must be well acquired and processed before feeding it into the neural network. Both stages, acquisition and processing, require a right amount of effort to execute and, most of the times, much more than training the network.

Lastly, the obtained model and results must be interpreted, verified and validated by following different techniques such as Cross-Validation (CV), Receiver Operation Characteristic (ROC) curve and other performance metrics. A decent examination of the recently referenced techniques will allow selecting a model that is well fitted to segment the PCG effectively.

A. Database

The PCG recordings, extracted from the so called PhysioNet Database [6], was utilized.

The Challenge 2016 carried out by the Computing of Cardiology (CinC) 2016 provided participants with a reasonably big dataset comprised of PCG and EKG recordings along with various annotations such as a patient identifier, abnormality of the signal and so on. Nevertheless, this challenge asked participants to classify PCG according to the pathology presented in them. Additionally, the trial provided a link to Springer's implementation of *Logistic Regression Hidden Semi-Markov Model* (LR-HSMM) for heart sounds segmentation and with it, 792 PCG recordings from 37 patients with R wave and end of T wave annotations. These wave annotations have correspondences in time with S1 and S2 in the PCG respectively, according to [2], and Springer *et al.* implemented a labelling algorithm in [3] to automatically generate labels used for training models. Furthermore, experts identified these recordings as healthy and unhealthy. The dataset maintains an equilibrium between normal and abnormal signals which is essential for training models for them to learn different features or patterns off of the data.

B. Annotations and labeling

In *Supervised Learning* extracting labels is a crucial step to go through. It can be performed manually or automatically. The former is commonly done by specialists in the field, and the later is fundamentally an algorithm responsible for computing them.

In this work, labels were extracted using a labelling algorithm provided by Springer [3]. This algorithm leverages annotations provided in the dataset corresponding to the EKG waves (R-wave and end of T-wave) and the homomorphic envelope [2] which is depicted in Figure 1. Based on each annotation, a lower and upper bound is defined to label S1 and S2. Between S2 and the S1 from the following cardiac cycle lies the diastolic interval and the is assumed to be the systolic interval.

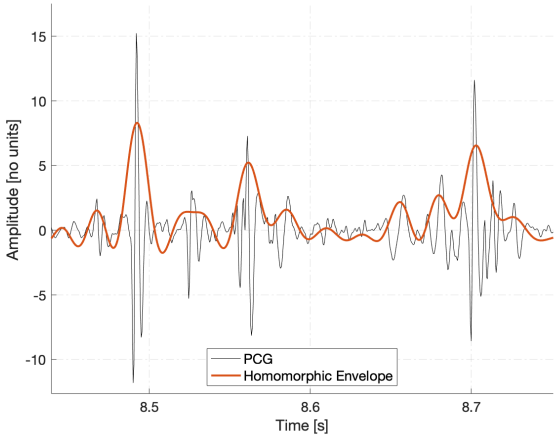

Fig. 1: Homomorphic envelope (normalized) used for PCG labeling. In red, a smoother signal, corresponding the envelope, and in black, the PCG.

It is worth mentioning that the previously mentioned algorithm should only be used offline. The training labels this algorithm yields, should only be used to train the neural network. It is not advisable to use this algorithm in real-time applications due to the dependency of the EKG signal. If so, performing PCG segmentation online could be quite troublesome. Moreover, the algorithm has to be tuned manually to retrieve reliable labels. Thus, the need to develop a method independent of EKG signals, and a trained deep neural network is a good way of solving this problem. Additionally, it can be implemented in real-time embedded systems with special care to perform an online segmentation, if that would be goal. An example of a automatically labelled PCG signal is illustrated in Figure 2

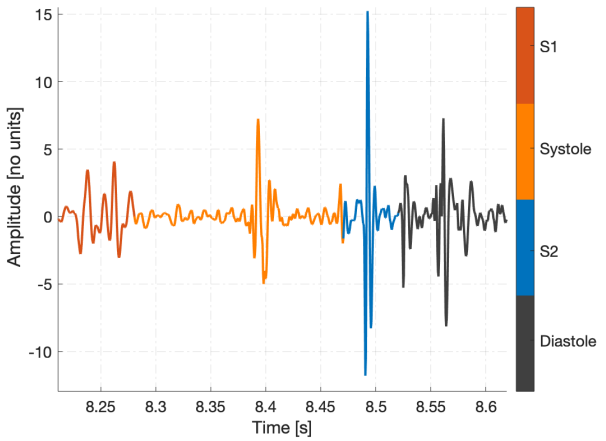

Fig. 2: Automatically labeled PCG (normalized). Four states are identified in four different colors, showing the beginning and ending of each one.

III. Approach and DRNN models

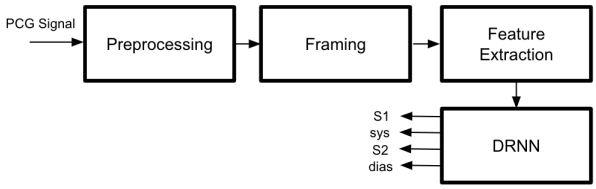

Fig. 3: Heart sounds segmentation approach.

A. Preprocessing

Neural networks cannot ingest data without being adequately transformed. Otherwise, these would not perform at their best in both training and testing stages. Classification techniques expect classifying observations into the right class by observing explanatory variables or features. Most of the times, these features are not on the same scale. Thus, *standarization* is an excellent technique to perform on the data before feeding it into the network.

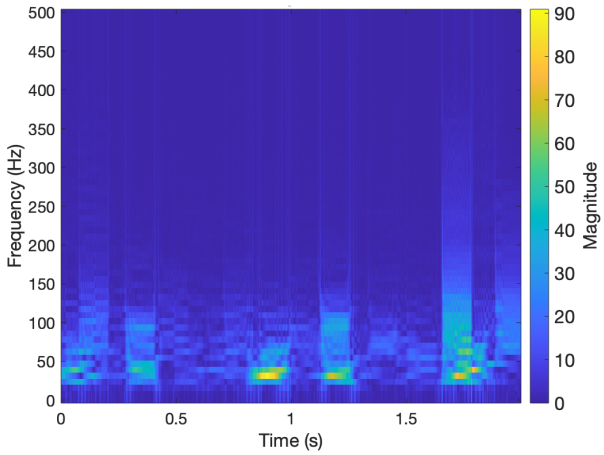

Fig. 4: Fourier Synchrosqueezed Transform across a portion of a PCG. It is possible to see the two main sounds (depicted in green), and both systole and diastole intervals, associated with higher and smaller energy levels, respectively.

$$Z_i = \frac{X_i - \mu_i}{\sigma_i}, \quad i = 1, \ldots, p \tag{1}$$

Where p is the number of features. Secondly, the frequency content in a PCG signal has been determined to be between 25 - 400 Hz in Schmidt work. However, we have identified that most of the frequency energy needed is contained in the 20 - 200 Hz range, as it is show in Figure 4.

B. Framing

Due to Deep Neural Networks (DNN) having a fixed input length and the signals not having the same duration, it is essential to perform a framing process on them.

N-dimensional patches were extracted from an N-dimensional signal $\boldsymbol{x}_k \in \mathbb{R}^T$ with a given stride τ for $k = 0, 1, \ldots, T-1$. These patches $\boldsymbol{z}_i$ are the inputs which the network is fed with. Thus, $\boldsymbol{z}_i \in \mathbb{R}^L$ is constructed by computing Equation (2).

$$\boldsymbol{z}_i = \begin{bmatrix} \boldsymbol{x}_{i\cdot\tau} \\ \vdots \\ \boldsymbol{x}_{i\cdot\tau+L-1} \end{bmatrix} \tag{2}$$

Where $N = \lfloor \frac{T-1-L}{\tau} \rfloor$ and L is the fixed length of the patches and $i = 0, 1 \ldots, N$ and $\lfloor x \rfloor$ denotes the greatest integer lower or equal than x. It is worth considering that after you select L you cannot frame signals whose length is lower than the selected length, therefore those signals will not be taken into account. Therefore, a good choice of L is equal to the lowest signal length.

Example 1. *We want to create patches of length $L = 2000$ from a signal whose length is $T = 35000$. After these two parameters are set up, we just need to define how much overlapping we want between patches, defined by parameter τ. This means if we set $\tau = 1000$, 50% of the samples in each patch will be repeated in the adjacent ones and if it is set to $\tau = 2000$, patches will not intersect. Suppose we choose $\tau = 1000$, then N is equal to 32, meaning that the signal will be framed into 32 patches. It is important to note that if $T - 1 - L$ is not multiple of τ some samples will be discarded due to the floor function $\lfloor \cdot \rfloor$.*

C. Feature extraction

Many features have been used to perform PCG segmentation. Mostly envelopes were used by Schmidt [2] and Springer [3]. The former used the so-called Homomorphic Envelope, and the latter added three more envelopes, such as the Discrete Wavelet (DWT) Envelope, the Power Spectral Density (PSD) Envelope, and the Hilbert Envelope.

Nevertheless, in this work, we proposed a different approach to accomplish PCG segmentation. Fourier Synchrosqueezed Transform (FSST) [7] is a technique based on Short-time Fourier Transform (STFT) that maps STFT frequencies into instantaneous frequencies of the signal at a given time t. After computing the FSST on a PCG signal, just a frequency range is extracted from it. Frequencies in the range of 20 - 200 Hz were kept.

Example 2. *To illustrate how extracted features from the FSST impact on the choice of the input layer size, consider a*

framed signal by the method in III-B yielding a patch length of 2000. As Figure 3 suggests, we have to extract the features from the given patches, but for the sake of this example we take into account only one patch. The FSST will compute time-frequency characteristics of the given patch providing a fixed amount of instantaneous frequencies. For instance, we will get a feature matrix $F \in \mathbb{C}^{q \times p}$, where q is number of computed frequencies and p the number of timestamps (the later matches the length of the patch). However, we might be interested in a range of frequencies, and by selecting those, we get a smaller fixed number of frequencies that we can feed the input layer with, and so the architecture of the input layer should be properly configured.

D. Neural Network

The proposed approach in this work is DRNNs specialized in time-series data. Knowledge about past and future times is a great feature to have for neural networks and Long Short-Term Memory (LSTM) excel in this subject. One shortcoming in RNNs is the *vanishing gradient problem*. Networks that are deep and have a large number of units tend to vanish the gradient when *Backpropagation Through Time* (BPTT) algorithm is computed. So LSTM has proved to be a robust solution to the vanishing gradient problem and, at the same time, keeping the well-known benefits of RNNs.

1) Long Short-Term Memory (LSTM) Block: The LSTM network is a type of RNNs. It consists of several memory blocks. Figure 5 reflects the operations taking place within each one of them at a specific layer.

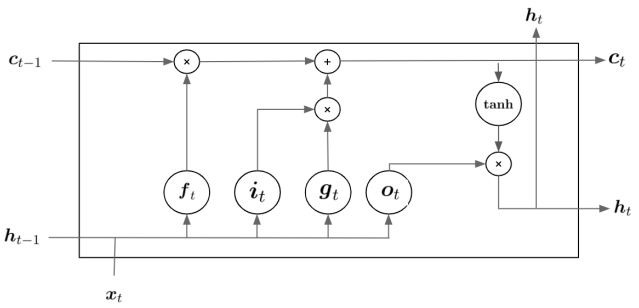

Fig. 5: Fundamental block in LSTM networks

For a given block and features $\boldsymbol{x}_t$ at a given time t in an LSTM layer, l, the following equations are computed:

$$\boldsymbol{i}_t^{(l)} = \sigma(\boldsymbol{W}_{ii}^{(l)}\boldsymbol{x}_t^{(l)} + \boldsymbol{b}_{ii}^{(l)} + \boldsymbol{W}_{hi}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{hi}^{(l)}) \quad (3)$$

$$\boldsymbol{f}_t^{(l)} = \sigma(\boldsymbol{W}_{if}^{(l)}\boldsymbol{x}_t^{(l)} + \boldsymbol{b}_{if}^{(l)} + \boldsymbol{W}_{hf}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{hf}^{(l)}) \quad (4)$$

$$\boldsymbol{g}_t^{(l)} = \tanh(\boldsymbol{W}_{ig}^{(l)}\boldsymbol{x}_t^{(l)} + \boldsymbol{b}_{ig}^{(l)} + \boldsymbol{W}_{hg}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{hg}^{(l)}) \quad (5)$$

$$\boldsymbol{o}_t^{(l)} = \sigma(\boldsymbol{W}_{io}^{(l)}\boldsymbol{x}_t^{(l)} + \boldsymbol{b}_{io}^{(l)} + \boldsymbol{W}_{ho}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{ho}^{(l)}) \quad (6)$$

$$\boldsymbol{c}_t^{(l)} = \boldsymbol{f}_t^{(l)} \odot \boldsymbol{c}_{t-1}^{(l)} + \boldsymbol{i}_t^{(l)} \odot \boldsymbol{g}_t^{(l)} \quad (7)$$

$$\boldsymbol{h}_t^{(l)} = \boldsymbol{o}_t^{(l)} \odot \tanh(\boldsymbol{c}_t^{(l)}) \quad (8)$$

Where in Figure 5, $\boldsymbol{h}_t$ is the hidden state at time t, $\boldsymbol{c}_t$ is the cell state at time t, $\boldsymbol{x}_t$ is the input at time t, $\boldsymbol{h}_{t-1}$ is the hidden state of the layer $t-1$ or the initial hidden state at time 0, and $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{g}_t$, $\boldsymbol{o}_t$ are the input, forget, cell and output gates, respectively. σ is the sigmoid function, and $\odot$ is the Hadamard product.

In a multi-layer scheme the hidden state $\boldsymbol{h}_t^{(l-1)}$ of a previous layer can be multiplied by a drop-out $\delta_t^{(l-1)}$ coefficient where each $\delta_t^{(l-1)}$ is a random Bernoulli variable with probability p.

LSTM cells decide whether to keep information from a previous time $t-1$ using the forget gate by taking into account $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$. Then using the input gate and the cell gate can choose what information is a candidate to be stored in the cell. Once the information has been chosen $\boldsymbol{c}_{t-1}$ is updated into the new state $\boldsymbol{c}_t$. Finally, the hidden state $\boldsymbol{h}_t$ is computed in Equation (8).

2) Bidirectional Long Short-Term Memory (BiLSTM): BiLSTM networks are an extension to LSTM, in which training is performed in both time directions simultaneously possible by using two embedded RNN layers. One backward and another one forward depicted in Figure 6. Both backward and forward hidden-states are then fed into the next layer. In some instances, it can be another BiLSTM layer. It is worth mentioning that each block comprises the computations described in Section III-D1.

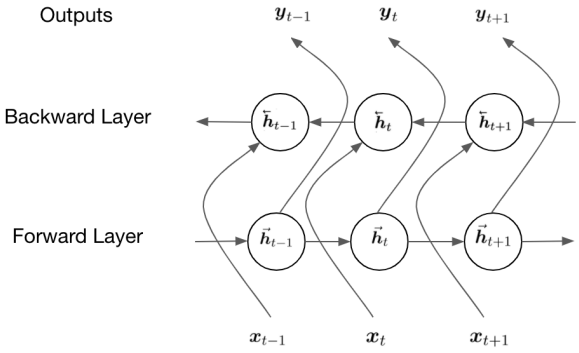

Fig. 6: BiLSTM network

Bidirectional Recurrent Neural Networks (BRNN) can also be built upon different RNN schemes such as Bidirectional Gated Recurrent Units (BGRU).

3) Architecture: The LSTM architecture in Figure 7 is comprised of three hidden layers besides the input and output layer. The input layer has a dimensionality of 44 correspondings to the features of interest. It is connected to a first 200-unit BiLSTM hidden layer activated by a ReLU function which is also connected to a second hidden layer with the same characteristics. Its output is then fed into a fully-connected or dense layer to compute the corresponding scores. Finally, a softmax layer is liable for computing the likelihood of belonging to a particular class. This architecture could seem straightforward and paltry, but it does the job classifying with a more than worthy performance.

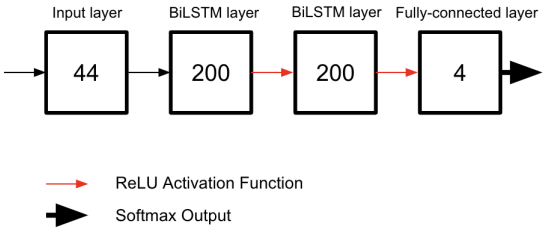

Fig. 7: Proposed DRNN.

4) Drop-out Regularization: Deep neural networks usually tend to be very deep. Convolutional Neural Networks (CNN) is one good example. Networks having a large number of hidden layers yields to have a large number of parameters to optimize. Nevertheless, the more layers a DNN has, the more likely it is to overfit. Suggesting that it will learn every detail off of the data trained with and it will lose generality.

Drop-out is a technique for addressing this problem. The central idea is to randomly drop units from the network during training and preventing neurons from co-adapting too much. In broad, drop-out is implemented by deactivating neurons in each iteration with a probability p. During the whole training process, all neurons will have been deactivated the same amount of times.

This technique significantly reduces overfitting and gives significant improvements over other regularization methods.

E. Model training & testing

Training was performed using cross-validation techniques, namely *K-Fold Cross Validation*. Training. Validation and test folds were deliberately chosen to begin training the neural network. With K being the number of folds which was picked to be 10. Along these lines, every observation was used to train and test the model. Say that the picked model is the one performing at its best on the testing dataset.

Most of the time contributed in training is used to choose the appropriate hyperparameters of the model. Generally, this is done by trial and error. The chosen hyperparameters that best performed are shown in Table I.

A gradient threshold is defined to avoid any issues with gradient explosions. Therefore, if the gradient in absolute value is greater than the threshold, it will be clipped. Another technique to avoid overfitting is using a validation set to control how close the validation loss is to the training loss called *Early Stopping*. The stopping is set by defining a *validation patience* which outlines how many times the validation loss can be higher than the minimum validation loss computed at a given iteration. If this criterion is met, then the training progress stops.

The optimizer is another option to set up. *Adaptive Moment Estimation* (ADAM) is a stochastic gradient-based optimization method to find the weights in a neural network. It requires that a learning rate is set and it can also be adaptive. It implies that every a fixed amount of epochs the learning rate decreases by some factor.

The hyperparameters in Table I were chosen with certain criterion. For instance, the mini-batch size, initial learning rate, the learn rate drop period and gradient threshold are the most common values in the literature, which were found to have the best results based on the architecture and data selected. Ultimately, the number of epochs generally are defined between 10-30, although via trial and error, after 6 epochs we noticed the performance of did not improve whatsoever if the network was trained for longer epochs, and in some cases, the network was prone to overfit. Thereby, 6 epochs seems a reasonable value to reduce the training time and the possibility of overfitting. Addtionally, a validation patience was added, and in this case chosen to be 6 because we have seen that the accuracy did not improve after 6 failures.

TABLE I: CHOSEN HYPERPARAMETERS

Hyperparameter	Value
Optimizer	ADAM
Epochs	6
Mini-Batch Size	50
Initial Learning Rate	0.01
Learn Rate Drop Period	3
Gradient Threshold	1
Validation Patience	6

IV. RESULTS

Performance of a model is a crucial step to select the appropriate model. Most common metrics reported are precision (P_+), sensitivity (Se), F1-score (F_1) and accuracy (ACC) computed based on true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$P_+ = \frac{TP}{TP + FP} \tag{10}$$

$$Se = \frac{TP}{TP + FN} \tag{11}$$

$$F_1 = 2\frac{P_+ \cdot Se}{P_+ + Se} \tag{12}$$

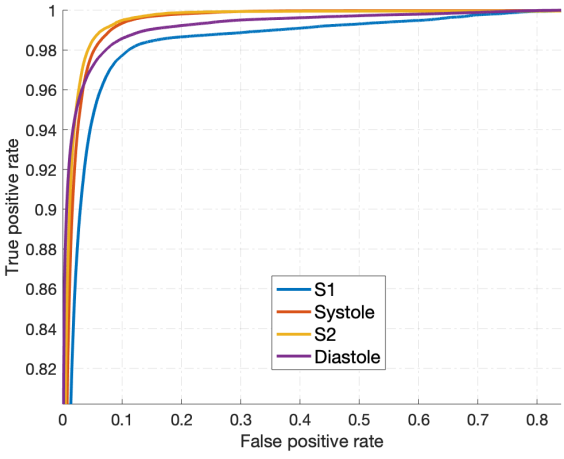

Fig. 8: Receiver Operating Characteristic curves. Curves corresponding to each class (S1, sys, S2, dias).

Another necessary result to report is the so-called ROC curve, computed off the scores from the last hidden layer and test labels. In Figure 8, a ROC curve is depicted for a given fold, which is constructed by plotting True Positive Rate (TPR) and False Positive Rate (FPR). TPR is also known as sensitivity defined in Equation (11) and FPR is also known as *fall-out* formulated in Equation (13).

$$FPR = \frac{FP}{FP + TN} \tag{13}$$

Once the ten models have been trained, one has to be selected. *Area Under the Curve* (AUC) is computed for each model in Table II, and the highest is picked. Since there are four classes, the average is computed and used for comparison.

TABLE II: AUC SCORES

K-Fold	AUC (%)
1	99.1
2	98.5
3	98.5
4	97.2
5	98.9
6	98.1
7	98.5
8	98.9
9	98.9
10	98.9

Lastly, Equations (9) to (12) are averaged across all trained models to report the final results in Table III.

TABLE III: FINAL RESULTS UPON CROSS-VALIDATION.

State	P_+ (%)	Se (%)	F_1 (%)
S1	85.7	86.5	86.0
Sys	90.0	90.0	90.0
S2	87.1	86.7	86.9
Dias	94.9	94.6	94.8
Average	89.3	89.5	89.4
ACC (%)	91.34		

TABLE IV: STATE-OF-THE-ART ALGORITHMS COMPARISON.

Algorithm	ACC (%)	P_+ (%)	Se (%)	F_1 (%)
BiLSTM	91.34	89.3	89.5	89.4
LR-HSSM [3]	92.52	95.92	95.34	95.63
CNN+HMM [4]	93.7	95.7	95.7	95.7

V. CONCLUSION

In this work, a novel heart sounds segmentation model has been presented. An LSTM model is accompanied by a time-frequency feature extraction procedure carried out by the Fourier Synchrosqueezed Transform (FSST). It is shown that choosing the right method to extract features and the right neural network architecture yields, in comparison to other proposals, an almost state-of-the-art performance as shown in Table IV. For instance, the modified U-Net neural network used by Renna *et al.* [4] with more than 20 layers.

Data augmentation and data addition can also be increased to make the network deeper and to train it for a more significant number of epochs. This means that using an LSTM network can achieve higher performances by developing a more complex architecture since these cells are specialized in time-series data.

A possible future extension from this approach is to implement a post-processing stage in order to imply correct transitions between states, which is the main limitation of this approach. Thus, improving the performance.

REFERENCES

[1] H. Liang, L. Sakari, and H. Iiro, "A heart sound segmentation algorithm using Wavelet Decomposition and reconstruction," in *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 'Magnificent Milestones and Emerging Opportunities in Medical Engineering' (Cat. No.97CH36136)*, vol. 4, Chicago, IL, USA, Oct 1997, pp. 1630–1633.

[2] S. E Schmidt, C. Holst-Hansen, C. Graff, E. Toft and JJ. Struijk, "Segmentation of heart sound recordings by a duration-dependent hidden Markov model," *Physiological measurement*, vol. 31, no. 4, pp. 513–529, Mar. 2010.

[3] D. B. Springer, L. Tarassenko, and G. D. Clifford, "Logistic Regression-HSMM-Based Heart Sound Segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, pp. 822–832, April 2016.

[4] F. Renna, J. H. Oliveira, and M. T. Coimbra, "Deep Convolutional Neural Networks for Heart Sound Segmentation," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–1, 2019.

[5] P. F. O. Ronneberger and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[6] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[7] T. Oberlin, S. Meignen, and V. Perrier, "The Fourier-based Synchrosqueezing Transform," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 05 2014, pp. 315–319.