Heartbeat Classification: A Comparison Between Deep Learning and Classical Machine Learning Approaches

Giovanni Bonfanti$^{1[3348A]}$

Università degli Studi di Milano
giovanni.bonfanti2@studenti.unimi.it

Abstract. Automatic recognition of anomalies in heart function based on the acoustic characteristics of the heartbeat is a very relevant research field. In this study, a comparison between deep learning and traditional machine learning approaches for heartbeat classification is proposed. Specifically, two deep learning models –Long short-term memory (LSTM) and Convolutional Neural Network (CNN)– with two classical machine learning models – K-nearest neighbors (KNN) and Random Forest (RF)– are used to classify the sounds and compared to understand their pro and cons. The heart sound dataset used includes five classese, one normal (N) and four anomalies, respectively, aortic stenosis (AS), mitral regurgitation (MR), mitral stenosis (MS) and mitral valve prolapse (MVP).

Keywords: Cardiovascular Desease · Heart sound classification · Audio Features extraction · Deep Learning

1 Introduction

According to the World Health Organization, the most deadly diseases are related to the cardiovascular system [1]. Early diagnosis can help the patient to receive treatment on time and to reduce the risks caused by these pathologies. Continuous monitoring can therefore help a lot, but an expert cannot always be present to provide an accurate diagnosis; however, tools such as smartphones can be used to perform a self-check and provide a first evaluation in detecting heart problems, or at the same time, these tools can help doctors to provide the correct diagnosis. The heart sound can be an indicator of cardiovascular problems. In a healthy person, the beat sound is formed by two distinct sound ("lub-dub"). The first sound is associated with the closing of the atrioventricular valve at the onset of the systole, and the second sound is associated with the closure of the semicircular valve (aortic valve and pulmonary valve) at the end of the systole. Physiological problems in the heart, like the ones considered in this study, can alter this sound, adding rustles caused by the flowing of the blood at wrong timing or the struggle of the blood to flow through narrow or damaged valves. The goal of this study is to do a comparison between existing classification algorithms, in terms of accuracy, complexity and explainability. The data used, come from the dataset [2], and are divided into five classes, with 200 recordings for each class. Each file consists of the record of 3 heartbeats, already filtered to maintain only the valuable information.

2 Dataset

The heart sound dataset (HSD1) used is a public dataset. It consists of 1,000 signal samples in wav format with a sampling frequency of 8 kHz. Each file is filtered and cut to maintain only 3 beats each and to remove unwanted sound like noise or sound associated with other muscular contraction like respiration. The dataset is divided in 5 classes (Fig. 1), consisting in one normal class (N) and four anomalies:
Aortic stenosis (AS) is when the aortic valve is too small. The typical sound of aortic stenosis is a high-pitched, "crescendo-decrescendo" murmur [3].

Mitral regurgitation (MR) is when the heart's mitral valve does not close properly, allowing blood to flow back to the heart.
Mitral stenosis (MS) is when the mitral valve is damaged and cannot fully open. The typical sound of Mitral stenosis is a loud first heart sound, an opening snap, and a low-pitched, rumbling diastolic murmur [4].
Mitral valve prolapse (MVP) is the prolapse of the mitral leaflets into the left atrium during systole.

The dataset is completely balanced among the classes, consisting in 200 records per class. Given the fact that the sounds in the dataset were very clear, a second dataset was created (HSD2) by adding random white noise to the signals, up to -22dB. The datasets were splitted in two subsets for the experiment: 80% used for the training and 20% for the testing of the algorithms.

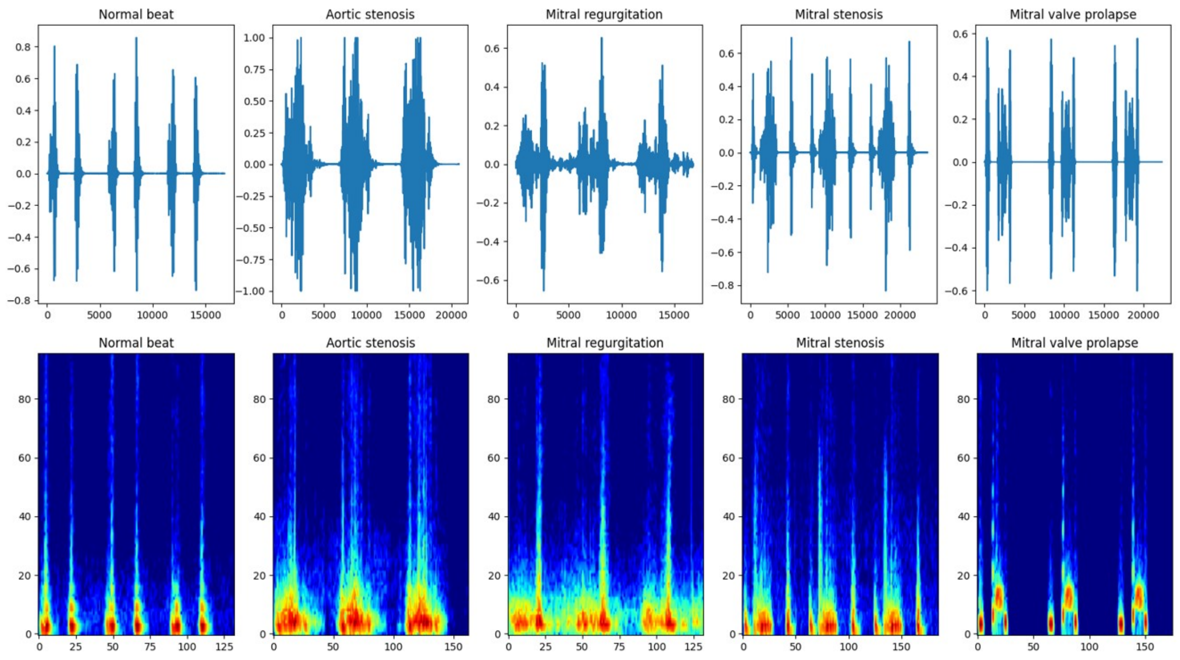


Fig. 1. Examples of the signals in the dataset

3 Method

3.1 Classical Machine learning approach

Classical machine learning algorithm used in this study uses traditional statistical learning tecniques. They classify data based on learned simple decision rules, but they cannot exploit high-level patterns in the data.

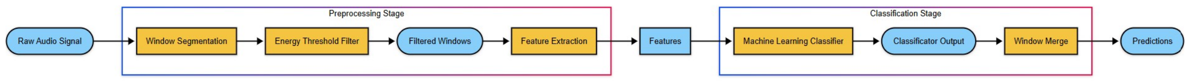


Fig. 2. Block diagram for the classical machine learning approach

3.1.1 Preprocessing stage

In this stage, the audio tracks are segmentated in fixed-length windows of 60ms, with an overlap of 50%. For each window a set of features is extracted using the librosa [5] library in python: Energy, Zero crossing rate, spectral centroid, spectral rolloff and mel-frequency cepstrum coefficients. Given the fact that silence between two heart beats is completly useless in the classification task, the energy computed on the windows is used to filter out silence segments. Specifically, the windows that have energy below a threshold T are simply ignored. The features for each window are then passed to the classification stage.

3.1.2 Classification stage

In this stage KNN and RF are used to classify the audio segments based on the features. Classification is performed on each window separately, and for each audio track, the final predicted label is computed as the trend of the predicted labels across all segments of the track.

KNN classifier works by computing the euclidean distance between the feature vector being classified and each feature vector of the training set as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2} \tag{1}$$

Where $\mathbf{x}$ is the feature vector being classified, $\mathbf{y}$ is the feature vector of a sample in trainin set, and n is the number of features.
The class predicted is the majority class of the k closest neighbors.

RF classifier works by building N decision tree during training, and outputs the class that is the majority vote of each individual tree. Each tree is trained on a random subset of the data, in this way the algorithm is able to generalize better compared to a single decision tree trained on all the data

3.2 Deep learning approach

Deep Learning is a subset of machine learning, that uses multilayered neural networks, called deep neural networks (DNN), to learn and understand patterns in data. Using high computational power and large datasets, DNN are able to understand high-level relationships and structures in data, allowing them to generalize and find the pattern also in new unseen data. In this work, two approaches were proposed, inspired by the work of [8]. The process for the classification of the signal is described in the fig.3. As shown, the process is divided in two parts: Preprocessing and Classification.

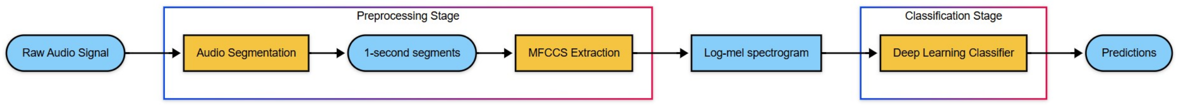


Fig. 3. Block diagram for the deep learning approach

3.2.1 Preprocessing Stage

The goal of this stage, is to extract and process the features useful for the classification part. The first step is to split all the audio signals in segment of 1 second. Each segment will then be considered as a indipendent file, ensuring that classification is consistent and not affected by variations in track length. Next, the only feature extracted is the log-mel spectrogram. This spectrogram is then treated as an image for the classification stage.

3.2.2 Classification stage

In this stage, the spectrograms are fed into a deep learning classification model, which has the goal to determine which class is associated with every audio track. Two different models have been tested: a LSTM fig.4 and a CNN fig.5.

LSTM is a special type of recurrent neural network layer. In this architecture, data are fed into LSTM layer one time sample at a time and the network is able to "memorize" and understand both long-term and short-term temporal dependencies in the data. In this implementation, 2 LSTM layers had been used to exploit this dependencies, then the output at the last time step of the second LSTM layer is fed into two fully connected layers wich perform the final classification.

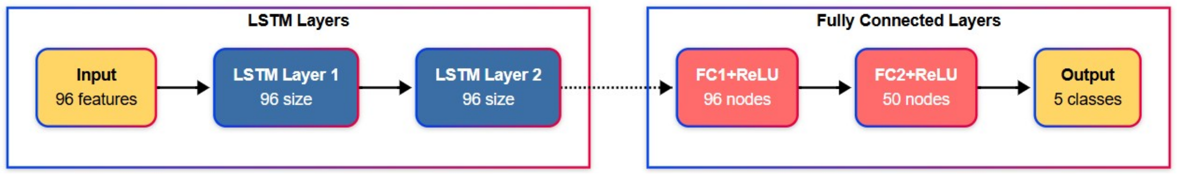


Fig. 4. LSTM block diagram

CNN is a type of neural network, particularly suitable for pattern recognition in images. In this implementation, the log-mel spectrogram is treated like an image fed into the network. The network is composed by 3 convolutional layers, that can find increasingly high-level shapes and patterns in the spectrogram, then the output of the third convolutional layer is fed into two fully connected layers wich perform the final classification.

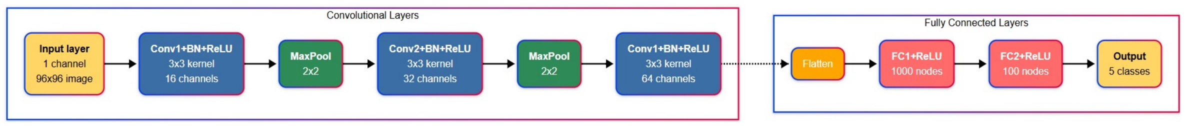


Fig. 5. CNN block diagram

3.3 Training details

All models were trained using a 80-20 train-test split, both for HSD1 and HSD2. For the implementation of KNN and RF the classes from scikit-learn [6] library were used. Only the number of neighbors (k=3) for KNN and the number of trees (n=100) for RF were explicitly specified; all other parameters were left at their default values. CNN and LSTM had been built using pyTorch [7], the hyperparameters used are shown in Table 1

Table 1. Hyperparameters used in Deep neural Networks

Hyperparameters	CNN	LSTM
Learning rate	6e-6	6e-6
Batch size	100	50
Number of epochs	500	2000

4 Results

The algorithms are confronted on 3 different topics: performance, complexity and explainability.

4.1 Performance

The performances of these algorithms are evaluated in terms of:

Precision : Is the measure of how many of the predicted positive were actually positive. It's defined as

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall : Is the measure of how many of the total positive were predicted as positive. It's defined as

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-score : Is the harmonic mean between precision and recall. The armonic mean is used to favoure balanced value. It's defined as

$$\text{F1-Score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Accuracy : Is the proportion between the correct predictions out of all predictions made. It's defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Table 2 shows the classification results obtained using the proposed models, both for the clean and the noisy dataset. For HSTD1, all the models performed in a similar way, with the best result achieved using the CNN model with a F1-score of 0.96. For HSTD2, on the other hand, CNN's performance remained quite high, while the performances of the other algorithms dropped, to an F1-score per KNN of 0.77. In the confusion matrices shown in Fig. 6 and in Fig. 7, each row represents the instances in a true class, and each column represents the instances in a predicted class. The correctly predicted instances are shown in the diagonal of the matrix, while the values outside the diagonal show the incorrectly predicted instances. In the Machine learning algorithm, the total number of sample used for testing was 40 for each class. In the deep learning algorithm on the other hand, the total number of samples is different for each class, but still very balanced. This is caused by the fact that before extracting the features, each audio track had been splitted in fixed-length segments.

Table 2. Performance metrics for different algorithms on two datasets

Dataset	Algorithm	Precision	Recall	F1-score	Accuracy
HSTD1	KNN	0.93	0.92	0.92	0.92
	RF	0.95	0.94	0.95	0.94
	CNN	0.96	0.96	0.96	0.96
	LSTM	0.95	0.95	0.95	0.95
HSTD2	KNN	0.82	0.78	0.77	0.78
	RF	0.85	0.84	0.84	0.84
	CNN	0.93	0.93	0.93	0.93
	LSTM	0.87	0.87	0.87	0.87

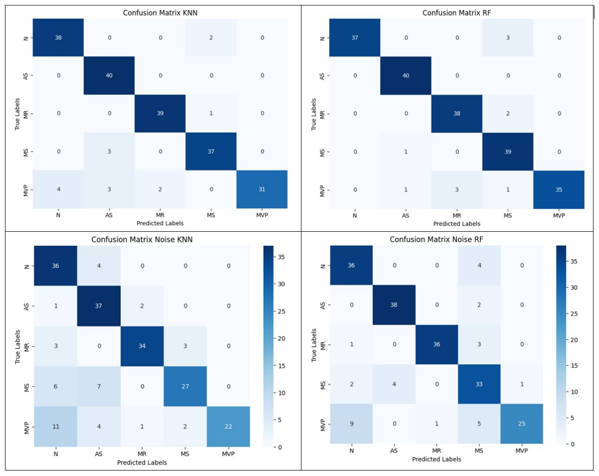


Fig. 6. Confusion matricies for Machine Learning Algorithms

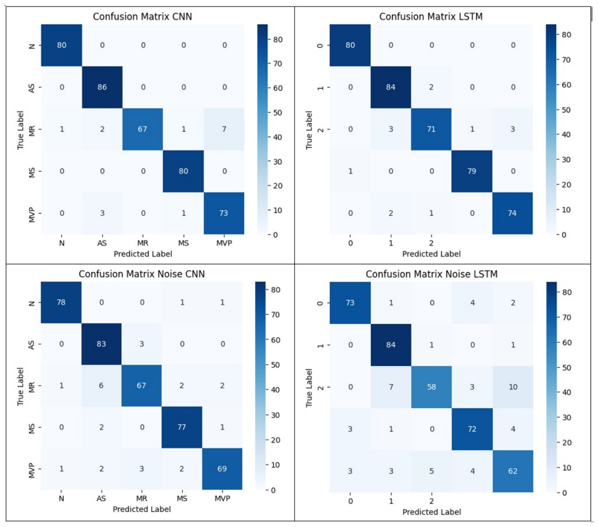


Fig. 7. Confusion matricies for Deep Learning Algorithms

4.2 Complexity

Complexity was evaluated in terms of Training time, Inference time and dimension of the model. For the Deep learning algorithms the training was done both on a CPU and on a GPU, to exploit parallel computations. CPU training was conducted on an *Intel Core i5-1135G7* with 8GB of RAM, while GPU training used an *NVIDIA T4* with 16GB of RAM.

Table 3. Complexity measures

Algorithm	CPU training	GPU training	Inference	Dimension
KNN	0.002 s	//	0.21 s	3 Mb
RF	13 s	//	0.02 s	100 Mb
CNN	95 m	4 m	0.03 s	10 Mb
LSTM	200 m	4 m	0.04 s	0.641 Mb

As shown in Table 3, among traditional machine learning algorithms, KNN had the fastest training time, despite having the longest inference time due to the need to compare each test sample with the entire training dataset. In contrast, RF had longer training time but faster inference time due to the fact that the computationally demanding phase is the construction of the decision trees. Concerning deep learning models, GPU training drastically reduced the training time to 4 minutes. Inference times, instead, were comparable with the other models. Regarding the size of the models, it heavily depends on the implementation and, for classical machine learning models, also on the size of the training set. In this specific experiment, the lightest model was LSTM, with 0.641Mb of storage required.

4.3 Explainability

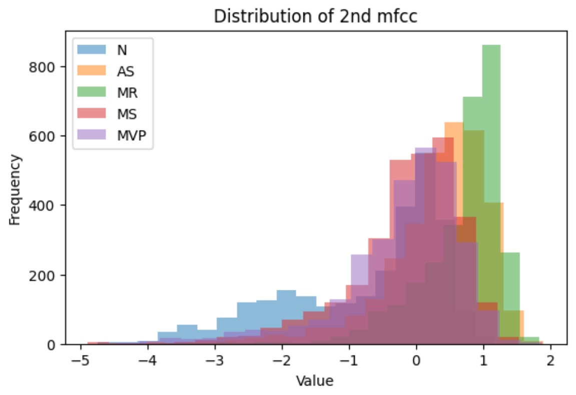
In addition to performance and complexity measures, the evaluation of the explainability of the model is very important in medical application, as it can support doctors in the final diagnosis

4.3.1 KNN The prediction process is very clear and interpretable. For each prediction is possible to know which saples were the nearest neighbours and wich features were the most discriminative in that prediction. For example, for the *sample 39* in the non-noisy dataset, the prediction was done in this way (with 0 indicating the Normal beat and 1 indicating Aortic Stenosis):

```
Test sample prediction: 0
Nearest neighbors (indices): [4803 20515 14823]
Labels of nearest neighbors: [0 1 0]
Distances to neighbors: [2.90506365 2.93277135 2.98252151]
```

4.3.2 RF The RF model allows the inspection of the decision trees, making it possible to find the most discriminative features. In this implementation, for example, with the non-noisy dataset, the most discriminative features were the 2nd and the 4th mfccs. As shown in the distributions showed in Fig. 8, these two fetures are not sufficient to separate the classes. RF will in fact still use all the features to do the classification

4.3.3 CNN For CNNs, is possible to visualize the output of the convolutional layers to understand wich parts of the spectrogram influenced more the classification. Each layer, however, gives in output many channels and every channel exploits a different caratteristic of the spectrogram. For this reason, it is still necessary to interpret the output of these layers, and this does not always lead to satisfactory results. The example in Fig. 9 shows the channel with higher activation for each layer for the *sample 20*



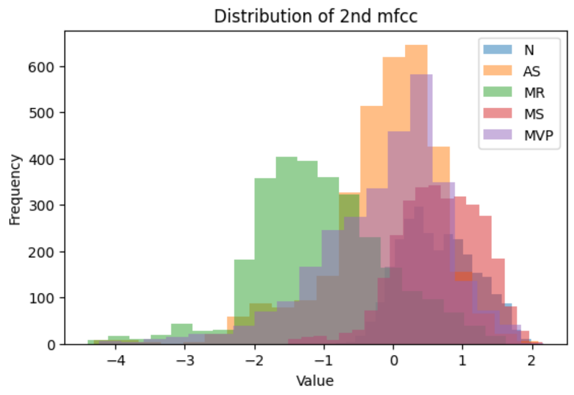


Fig. 8. Distribution of the 2nd and 4th mfccs for the non noisy dataset

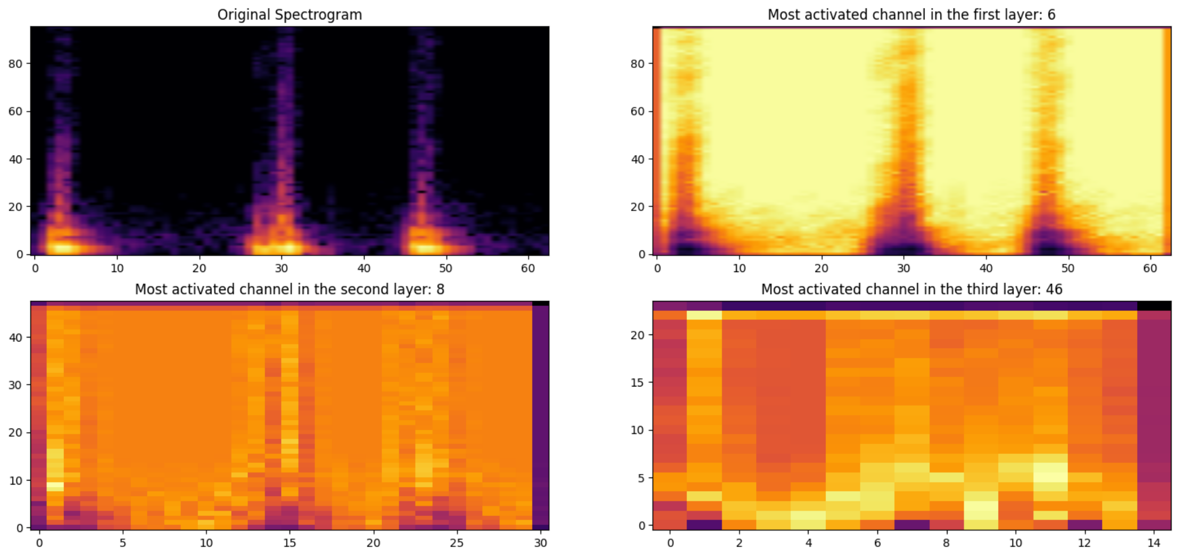


Fig. 9. Most activated channels of the layers of cnn

4.4 LSTM

While for the LSTM there are lots of tecniques to try to explain the decisions, their application in audio classification is complex and often brings results that are still hard to interpret. For these reasons, explainability results for LSTM are not reported in this paper.

5 Conclusions

This study compared 4 different algorithms for the classification of the Heart Beat sound. The objective was to evaluate the algorithms not only in terms of their performance, but also on complexity and explainability.

The results showed that the deep learning models achieved the highest classification performances on both datasets, since they were able to understand higher level patterns in the features. On the other hand, these models lack sufficient explainability, which limits their support for experts in critical decision-making contexts where model transparency is essential.

Classical machine learning algorithms also achieved very good performance results using the clean dataset, but their performance deteriorated considerably when applied to the noisy dataset. Their transparency, however, can be an essential factor in certain scenarios where understanding the reason for a decision is crucial.

In terms of algorithm complexity, deep learning models required significantly more training time than the other algorithms. However, since training only needs to be performed once, this is not a major concern. Regarding inference time, KNN requires significantly longer than the other algorithms. In this implementation, the inference time was even longer than the length of the window considered, making KNN unsuitable in real time scenarios. Notably, the LSTM model demonstrated the smallest model size, making it a suitable candidate for deployment in memory-constrained environments such as embedded systems. However, model size is strongly influenced by the implementation and, in some cases, the size of the training set. As a result, different configurations may lead to different model sizes.

This analisis was limited to a specific dataset and model configurations. Future work could involve larger and more diverse datasets, or parameter optimization. Furthermore, investigating capability of these algorithms in real-time scenarios, could be usefull in clinical context.

References

1. "The top 10 causes of death" https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death
2. “Classification-of-Heart-Sound-Signal-Using-Multiple-Features.” https://github.com/yaseen21khan/Classification-of-Heart-Sound-Signal-Using-Multiple-Features-/find/master.
3. "Aortic Stenosis - Physical Exam" https://www.healio.com/cardiology/learn-the-heart/cardiology-review/topic-reviews/aortic-stenosis/physical-examination.
4. "Mitral Stenosis" https://www.msdmanuals.com/professional/cardiovascular-disorders/valvular-disorders/mitral-stenosis.
5. McFee, Brian Raffel, Colin Liang, Dawen Ellis, Daniel Mcvicar, Matt Battenberg, Eric Nieto, Oriol. (2015). librosa: Audio and Music Signal Analysis in Python. 18-24. 10.25080/Majora-7b98e3ed-003.
6. and Bertrand Thirion and Olivier Grisel and Mathieu Blondel and Andreas Müller and Joel Nothman and Gilles Louppe and Peter Prettenhofer and Ron Weiss and Vincent Dubourg and Jake Vanderplas and Alexandre Passos and David Cournapeau and Matthieu Brucher and Matthieu Perrot and Édouard Duchesnay. Scikit-learn: Machine Learning in Python
7. Adam Paszke and Sam Gross and Francisco Massa and Adam Lerer and James Bradbury and Gregory Chanan and Trevor Killeen and Zeming Lin and Natalia Gimelshein and Luca Antiga and Alban Desmaison and Andreas Köpf and Edward Yang and Zach DeVito and Martin Raison and Alykhan Tejani and Sasank Chilamkurthy and Benoit Steiner and Lu Fang and Junjie Bai and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library
8. Minh Tuan Nguyen, Wei Wen Lin, Jin H. Huang. Heart Sound Classification Using Deep Learning Techniques Based on Log-mel Spectrogram