Heart Murmur Detection from Phonocardiogram Recordings: George B. Mood PhysioNet Challenge 2022

Enoch Guo
The Stony Brook School, New York

Abstract

We present a machine learning method for heart murmur detection from phonocardiogram (PCG) recordings. Our approach consists of three steps: (1) Split recordings into 3-second waveforms; (2) Transform one-dimensional waveforms into two-dimensional time-frequency heat maps using Mel-Frequency Cepstral Coefficients (MFCC); (3) Classify MFCC using deep convolutional neural networks (CNN). We also use denoising autoencoder classification to improve the basic CNN.

1. Introduction

The goal of the Challenge is to identify the *present*, *absent*, or *unknown* cases of murmurs and the *normal* vs. *abnormal* clinical outcomes from heart sound recordings collected from multiple auscultation locations on the body using a digital stethoscope.

The Challenge data contain one or more heart sound recordings for 1568 patients as well as routine demographic information about the patients from whom the recordings were taken. The Challenge labels consist of two types:

1. *Murmur*-related labels indicate whether an expert annotator detected the *presence* or *absence* of a murmur in a patient from the recordings or whether the annotator was unsure (*unknown*) about the presence or absence of a murmur.
2. *Outcome*-related labels indicate the *normal* or *abnormal* clinical outcome diagnosed by a medical expert.

The Challenge data is organized into three distinct sets: training, validation, and test sets. The organizers have publicly released 60% of the dataset as the training set of the 2022 Challenge, and have retained the remaining 40% as a hidden data for validation and test purposes.

Therefore, we are given the training set that contains 3163 recordings from 942 patients. The public training set contains heart sound recordings, routine demographic information, *murmur*-related labels (*presence*, *absence*, or *unknown*), *outcome*-related labels (*normal* or *abnormal*), annotations of the murmur characteristics (location, timing, shape, pitch, quality, and grade), and heart sound segmentations. The private validation and test sets only contain heart sound recordings and demographic information.

2. Split into 3-second waveforms

The PCG recordings have different lengths, varying from 20608 to 258048 data points; with daterate being 4000 samples/sec, they vary from 5 seconds to about one minute.

PCG consists of cycles; each cycle consists of four states: S1, S2, systole and diastole. We split each recording into 3-second long overlapping segments, which is long enough to determine abnormality of heart sound. Each split wave consists of 12000 data points. In the following figure, we show a wave segment; to see the waveform better, we show a 1-second segment, instead of 3-second.

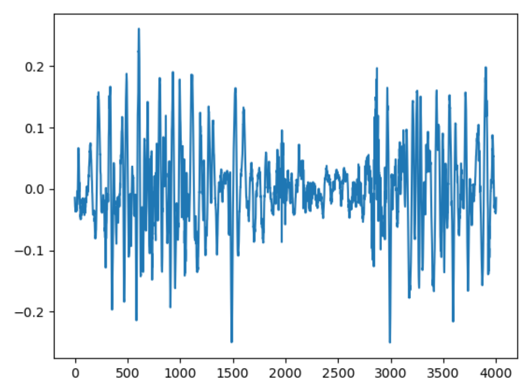

Figure 1. 1-second PCG Waveform

3. MFCC

Mel-Frequency Cepstral Coefficients (MFCC) [1] has been widely used in speech recognition. For each of the 3-second waveforms, we extract 4 frequency bands. If we extracted more frequency bands, other frequency bands would be mostly empty. Thus each one-dimensional wave (12000 long) is transformed into a two-dimensional heat map, of size $(4, 201)$, using `win_length` $= 100$ and `hop_length` $= 60$.

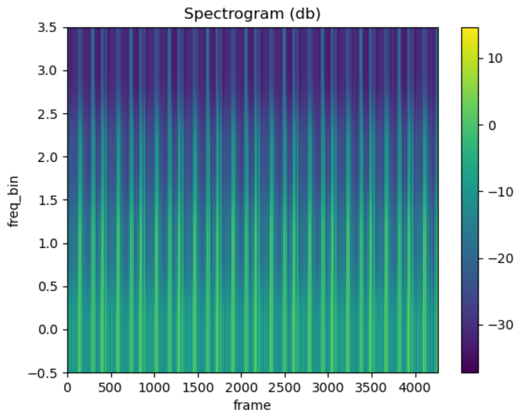

Figure 2. Log Scale MFCC

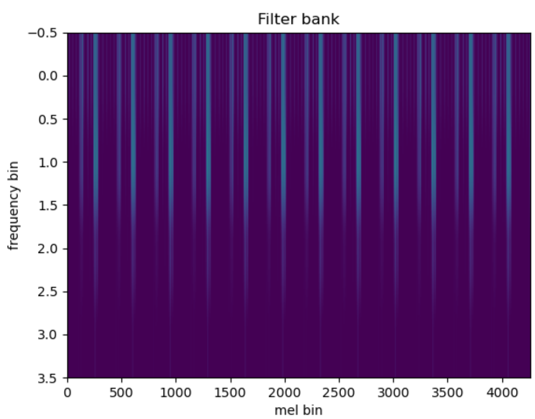

Figure 3. MFCC

4. Basic CNN

We treat the two-dimensional heat map as a two-dimensional image, and apply two CNNs: one for the *murmur* model and one for the *outcome* model. Let X denote the input and Y denote the output label. Our model can be described in the following figure,

$$\text{Input} \rightarrow \text{Label}$$

$$X \xrightarrow{Encode} Z \xrightarrow{Classify} Y$$

where X denotes the inputs, which are the two-dimensional images; Y denotes the output label, which is $(Present, Unknown, Absent)$ for the *murmur* model, and $(Abnormal, Normal)$ for the *outcome* model; Z is called the latent layer. For both models, we use three layers of convolution in CNN.

Both models look like:

1. *Encode*:
   (a) Conv2d
   (b) BatchNorm2d
   (c) ReLU
   (d) Conv2d
   (e) BatchNorm2d
   (f) ReLU
   (g) Conv2d
   (h) BatchNorm2d
   (i) ReLU
   (j) Flatten
2. *Classify*:
   (a) Dense
   (b) ReLU
   (c) Dense
   (d) Softmax

5. Denoising Autoencoder Classification

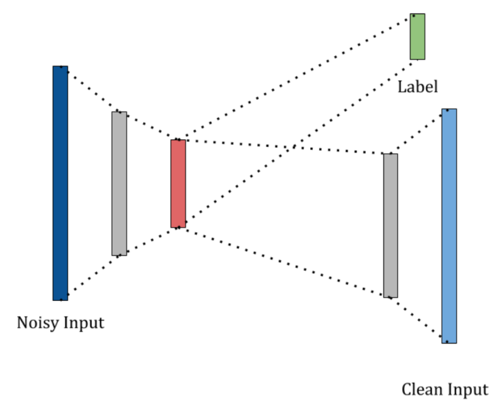

Figure 4. Denosing Autoencoder Classification Model

$$X' \xrightarrow{Encode} Z \quad \begin{matrix} \nearrow^{Classify} & Y \\ \searrow_{Decode} & X \end{matrix}$$

Training algorithm: Let X be input points (cyan), of size $(C, W, H) = (1, 4, 201)$, and Y be the classification labels (green).

1. *Add noise*: $X' = add\_noise(X)$. X' is blue. We have two methods to add noise: *blackout* and *random*. The *blackout* method chooses a small sample points (10%) to zero out their values; the *random* method changes the values by randomly with $\mu = 0$ and a small σ, say, 0.1.

2. *Denoising autoencoder decode*: Let Z be the latent layer (red). We train $X' \to Z \to (Y, X)$, where $X' \to Z$ is the *Encode* step above; $Z \to Y$ is the *Classify* step; and $Z \to X$ is the *Decode* step. In this step, the loss function for classify is zero. After *denoising autoencoder decode*, the model is called *regularized*.
3. *Classify from the regularized model*: continue to train the model $X' \to Z \to (Y, X)$ using cross entropy loss function for *classify*: $Z \to Y$.

In addition to the steps *Encode* and *Classify*, denoising autoencoder classification uses the *Decode* step, which is the inverse function of the corresponding *Encode* step, in the reverse order.

3. *Decode*
   (a) Unflatten
   (b) ConvTranspose2d
   (c) BatchNorm2d
   (d) ReLU
   (e) ConvTranspose2d
   (f) BatchNorm2d
   (g) ReLU
   (h) ConvTranspose2d
   (i) Sigmoid

6. Results

Our models are to classify 3-second waves. To classify the patients, we use the mean values of the probabilities for their waves.

Murmur scores:

```
AUROC=0.803
AUPRC=0.671
F-measure=0.607
Accuracy=0.817
Weighted Accuracy=0.624
Cost=20202.851
```

Outcome scores:

```
AUROC=0.616
AUPRC=0.624
F-measure=0.591
Accuracy=0.592
Weighted Accuracy=0.557
Cost=15119.790
```

Murmur scores (per class):

```
Classes,Present,Unknown,Absent
AUROC,0.788,0.792,0.830
AUPRC,0.613,0.481,0.919
F-measure,0.560,0.364,0.897
Accuracy,0.438,0.333,0.962
```

Outcome scores (per class):

```
Classes,Abnormal,Normal
AUROC,0.615,0.616
AUPRC,0.635,0.612
F-measure,0.580,0.603
Accuracy,0.541,0.647
```

7. Code

Our code can be found at the following link:
https://github.com/ejguo/physionet2022

References

[1] T. Ganchev, N. Fakotakis, G. Kokkinakis, "Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task," *Proc. of the SPECOM-2005*, October 17-19, 2005. Patras, Greece. Vol. 1, pp.191-194.

[2] Jordan Docter, Richard Gong, "Using Denoising Autoencoders in Multi-Task Learning to Regularize Classification," *http://github.com/jdocter/Denoising-Autoencoder -Classification/blob/master/ DenoisingAutoencoderClassification_2019.pdf.*