



Deep Learning

Heartbeat Sound Classification

U Kang
Seoul National University



In This Lecture

- Heartbeat sound
- Classifying heartbeat anomalies from audio
- Implement the convolutional neural networks using tensorflow



Outline

- ➡ ☐ **Problem Definition**
- ☐ Preprocessing Codes
- ☐ Answers



Motivation

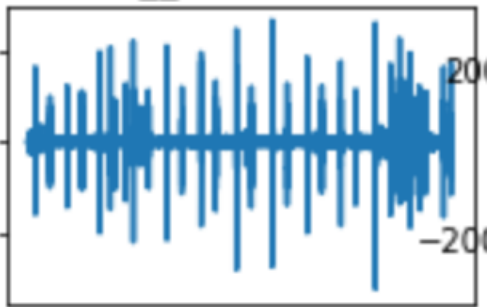
- According to the World Health Organization, cardiovascular diseases (CVDs) are the number one cause of death globally
- Detecting heart disease could have a significant impact on world health



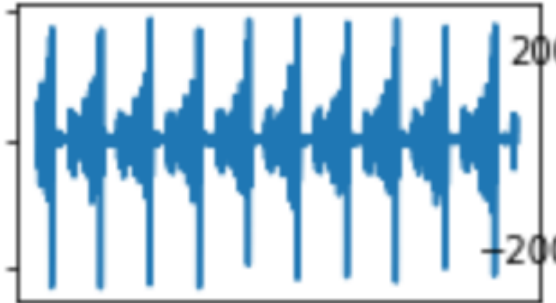
Goals

- Classify real heart audio (also known as “beat classification”) into one of three categories
 - one category is normal, and others are abnormal

Normal



Murmur



Artifact





Problem Definition

- **Given** heartbeat audio data
- **Classify** the data into the correct categories



Dataset (1)

- Heartbeat sound data collected from the general public via an iPhone app
- There are three categories
 - Normal (class 0)
 - Murmur (class 1)
 - Artifact (class 2)



Dataset(2)

- There are 176 heartbeat examples
 - Normal: 58, Murmur: 65, Artifact: 53
- The audio files are of varying lengths, between 1 second and 30 seconds
 - In this practice, we set the length of the heartbeat sound data to 1551 by downsampling the data



Category description

- Normal
 - Healthy heart sounds
- Murmur
 - There is a noise in a heart sound
- Artifact
 - There are a wide range of different sounds, including feedback squeals and echoes, speech, music and noise



Data File description

- Time series data (set_a/.wav)
 - Wav file of heartbeat sound
 - File name is (category)_(generated date).wav
 - The categories in file name can be different from the labels in input.csv file
 - The labels are re-labeled by an expert
- Meta data (input.csv)
 - Labels information of wav files.



Selection of an Algorithm

- The dataset has time-series structure
- We use Convolutional Neural Networks



Model Structure

- Implement 4 cnn layers and 1 fully-connected layer
- Use dropout in the fourth hidden layer
- Optimizer: Adam
- Loss function: cross entropy



Outline

☒ Problem Definition

 ☐ **Preprocessing Codes**

☐ Answers



Import libraries

- Import the libraries such as tensorflow, numpy, and so on

```
import numpy as np
import pandas as pd
import tensorflow as tf

from scipy.io import wavfile
from scipy.signal import decimate

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

```
INPUT_LIB = './data/heartbeat/'
SAMPLE_RATE = 44100
CLASSES = ['artifact', 'normal', 'murmur']
CODE_BOOK = {x:i for i,x in enumerate(CLASSES)}
NB_CLASSES = len(CLASSES)
```



Loading the Dataset (1)

- Using the pandas library, we prepare the input data
- Define functions to load the files

```
def load_wav_file(name, path):  
    _, b = wavfile.read(path + name)  
    assert _ == SAMPLE_RATE  
    return b
```

```
def repeat_to_length(arr, length):  
    """Repeats the numpy 1D array to given length, and makes datatype float"""  
    result = np.empty((length, ), dtype = 'float32')  
    l = len(arr)  
    pos = 0  
    while pos + l <= length:  
        result[pos:pos+l] = arr  
        pos += l  
    if pos < length:  
        result[pos:length] = arr[:length-pos]  
    return result
```



Loading the Dataset (2)

- Using the pandas library, we prepare the input data
- Read the csv file
- Read the wav files and convert those into 1-D array

```
df = pd.read_csv(INPUT_LIB + 'input.csv')
df['time_series'] = df['file_name'].apply(load_wav_file,
                                          path=INPUT_LIB + 'set_a/')
df['len_series'] = df['time_series'].apply(len)
MAX_LEN = max(df['len_series'])
df['time_series'] = df['time_series'].apply(repeat_to_length,
                                          length=MAX_LEN)
```




Loading the Dataset (3)

- You can check the contents:
 - Type `df.head()`

```
df.head()
```

	index	file_name	labels	time_series	len_series
0	0	artifact__201012172012.wav	0	[1.0, -3.0, -1.0, -7.0, -9.0, -2.0, -6.0, -5.0...	396900
1	1	artifact__201105040918.wav	0	[-2.0, 3.0, -4.0, 4.0, -3.0, 2.0, -1.0, 0.0, 0...	396900
2	2	artifact__201105041959.wav	0	[6.0, -4.0, -9.0, -1.0, -4.0, 1.0, -5.0, 2.0, ...	396900
3	3	artifact__201105051017.wav	0	[-85.0, -198.0, -214.0, -173.0, -177.0, -206.0...	396900
4	4	artifact__201105060108.wav	0	[53.0, -35.0, 47.0, 170.0, 340.0, 436.0, 535.0...	396900



Dividing the Dataset

- We prepare the training and test data
- Use 25% of the total data as test data

```
x_data = np.stack(df['time_series'].values, axis=0)  
y_data = pd.get_dummies(df['labels'].values.tolist()).values
```

```
x_train, x_test, y_train, y_test, train_filenames, test_filenames = \  
    train_test_split(x_data, y_data, df['file_name'].values, test_size=0.25)
```



Downsampling the Dataset

- Down-sample the data with what is in effect a very aggressive low pass filter.
- This is not needed for computational time, but it seems to improve generalization on this dataset.
 - The reason this works is probably that what you hear in the stethoscope is almost exclusively low frequency sounds, especially murmurs.

```
x_train = decimate(x_train, 8, axis=1)
x_train = decimate(x_train, 8, axis=1)
x_train = decimate(x_train, 4, axis=1)
x_test = decimate(x_test, 8, axis=1)
x_test = decimate(x_test, 8, axis=1)
x_test = decimate(x_test, 4, axis=1)
```



Preparing the Dataset

- Scale each observation to unit variance
- To use CNNs, we increase a dimension of data

```
x_train = x_train / np.std(x_train, axis=1).reshape(-1,1)  
x_test = x_test / np.std(x_test, axis=1).reshape(-1,1)
```

```
x_train = x_train[:, :, np.newaxis]  
x_test = x_test[:, :, np.newaxis]
```

```
print(f"X train shape: {x_train.shape}")  
print(f"X test shape: {x_test.shape}")
```

```
X train shape: (132, 1551, 1)  
X test shape: (44, 1551, 1)
```




Helpful Modules

- You may need these modules:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Conv1D
from tensorflow.keras.layers import MaxPool1D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.callbacks import EarlyStopping
```



Outline

- ☒ Problem Definition
- ☒ Preprocessing Codes
-  ☐ **Answers**



What You Need to Know

- Time series classification
- Convolutional neural networks



Questions?