Audio Data Analysis Using Machine Learning

Submitted By:
Trang Thi Huyen Nguyen

Supervised By:
Dr. Muhammad Fahim

[Summer Internship Report 2019]

Table of Contents

Abstract

Heart sounds have long been recognized as an additional diagnostic tool to indicate cardiovascular diseases. Using recordings from 2016 PhysioNet/CinC Challenge, the problem of identifying abnormal sounds from normal ones has been addressed by two approaches both from Machine Learning: 1) K-Nearest Neighbor (kNN), and 2) Convolutional Neural Network (ConvNet/CNN). The main features for these models are Mel Frequency Cepstral Coefficients (MFCCs). Implementation is written in Python3.

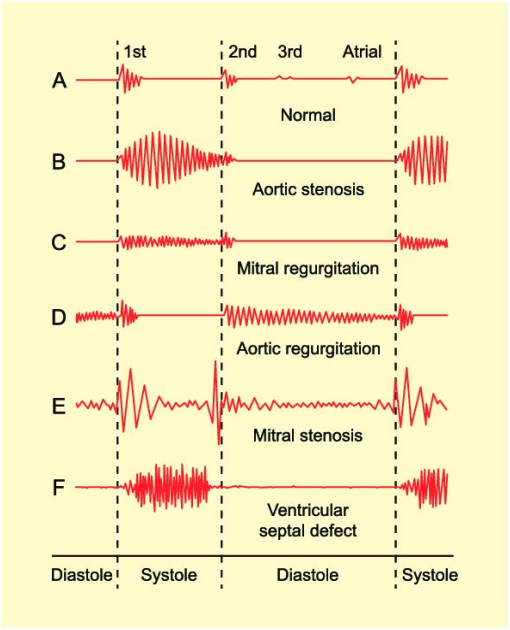
Introduction

According to WHO, cardiovascular diseases (CVDs) are the number 1 cause of death with about 17.9 million people died in 2016 (31% of all global deaths). As a diagnostic test for CVDs, heart sound signals (i.e., phono-cardiograms or PCGs) have been widely studied over the past few decades.

To autonomously classify normal/abnormal heart sounds, this study uses two Machine Learning techniques, i.e., K-Nearest Neighbor and Convolutional Neural Network with code written in Python 3 and dataset retrieved from 2016 PhysioNet/CinC Challenge. Dataset includes 3,240 .wav files with length from 5s to 120s and only 1 PCG lead which are recorded in uncontrolled environment (i.e., records corrupted by various noise sources like talking, stethoscope motion). Mel Frequency Cepstral Coefficients (MFCCs) are employed as features for both kNN and CNN models. Besides, spectral amplitudes and wavelet features are extracted and used in kNN model independently and with MFCCs for the purpose of comparing. F1 score is used as a primary metric.

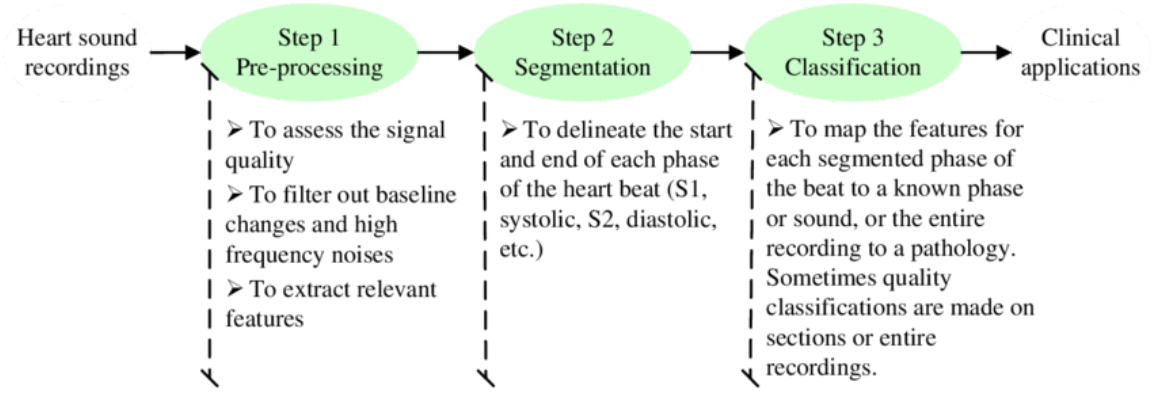
Related Work

Usually, Fundamental Heart Sounds (FHSs) include the first (S1) heart sound and the second (S2) heart sound. S1 is the beginning of isovolumetric ventricular contraction. Its frequency ranges from 10Hz to 140Hz (energy concentration in range of 25-45Hz). S2 is the beginning of diastole with closure of aortic and pulmonic valves with frequency from 10 to200Hz (energy concentration in range of 55-75Hz). Besides, there are other sounds such as the third (S3) heart sound, the fourth (S4) heart sound (20-70Hz); systolic ejection click (EC); mid-systolic click (MC); diastolic sound or



opening snap (OS); heart murmurs (diverse frequency ranges); and respiration (200-700Hz).

In clinical applications, automated analysis of the heart sound usually consists of three steps: 1) preprocessing (to filter out noises and extract relevant features); 2) segmentation (to identify the start and end of each phase of the heart cycle, namely S1, systolic, S2, diastolic); and 3) classification (to map features for each segmented phase). In the last few decades, automated segmentation and classification methods have been widely studied.



Typical segmentation methods can be divided into 4 classes: envelope-based, feature-based, machine learning, and Hidden Markov model methods (HMM). In envelope-based methods, a variety of techniques to construct envelops of heart sounds is used such as wavelet decomposition, Hilbert transform, cardiac sound characteristic waveform (CSCW), and simple squared-energy envelope. Frequency-, amplitude-, and entropy-based features are used in feature-based segmentation methods. Neural network and k-nearest neighbor are widely used as typical Machine Learning methods. Extended HSMM method by Springer et al., one of HMM-based methodology, is regarded as the state-of-the-art method in heart sound segmentation.

Artificial neural network-based (ANN), support vector machine-based (SVM), HMM-based, and clustering-based are four main categories of heart sound classification. Wavelet-based, energy spectrum, time-frequency features are usually employed as input for these models.

Methodology

Dataset is retrieved from the training set of 2016 PhysioNet/CinC Challenge. These heart sounds are collected in uncontrolled environment from different locations such as aortic area, pulmonic area, tricuspid area, and mitral area; and from both children and adults; normal people and patients with confirmed cardiac diagnosis. These records last from 5s to 120s and are placed in 6 folders from training-a to training-f with 3,240 .wav files in total. In this project, 70% of them is used for training and 30 % is used for test in kNN

model. For CNN, they are split into three sets - training set (70%), validation set (15%), and test set (15%). Since the dataset is biased (i.e., there is much more normal heart sounds than abnormal ones), F1 score is used as a primary metric besides accuracy, precision and recall.

Summary of training set used in 2016 PhysioNet/CinC Challenge			
Database	Abnormal	Normal	Total
Training-a	292	117	409
Training-b	104	386	490
Training-c	104	7	31
Training-d	28	27	55
Training-e	183	1958	2141
Training-f	34	80	114
Total	665	2575	3240

Preprocessing

Since lengths are different from each recording and the shortest one is 5s, only the first 5s from each record is considered. The 4^{th} order Butterworth Filter with bandpass between 25 and 400Hz can be used to remove spikes.

Feature extraction

Mel Frequency Cepstral Coefficients (MFCCs), which were introduced by Davis and Mermelstein in the 1980's, are considered as state-of-the-art feature in audio signal processing. In this project, they are used as main features. They are commonly derived in following steps:

Step 1: *Frame the signal into short frames (20-40ms)*. So that, in each frame, audio signal does not change much. However, if the frame is much shorter, samples are not enough to get a reliable spectral estimate. Frame step is usually something like 10ms (overlap is allowed). In next steps, for each frame, a set of 12 MFCCs will be extracted.

Step 2: *Calculate periodogram estimate of power spectrum for each frame*. This is motivated by human cochlea (an organ in human ear) to identify which frequencies are present in the frame.

Firstly, take Discrete Fourier Transform of the frame using formula:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \qquad 1 \leq k \leq K$$

Where $s_i(n)$ is i-th time domain signal (frame), range over number of samples in the frame; $h(n)$ is an N sample long analysis window (e.g. hamming window), and K is the length of the DFT.

Periodogram estimate of the power spectrum for this frame is calculated by:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

Generally a 512 point FFT is performed but and only the first 257 coefficients are kept.

Step 3: *Perform Mel filterbank to the power spectra, sum the energy in each filter*. Since the cochlea cannot distinguish between two closely spaced frequency and this effect becomes significant as frequencies increase, the Mel filterbank is applied to get the idea of how much energy in various frequency regions. Mel scale is used to space and make the filterbanks.

First, Mel-spaced filterbank - a set of 20-40 (26 is standard) triangular filters of length 257 (form step 2) is calculated using the formula:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

where M is the number of wanted filters, and $f()$ is the list of M+2 Mel-spaced frequencies.

Then, they are multiplied with the power spectrum. After that, the coefficients are added up to give an indication of how much energy is in each filterbank.

Step 4: *Take the logarithm of all filterbank energies*. This compression makes the features match more closely what humans hear because humans don't hear loudness on a linear scale (generally, perceived volume of a sound double if it's energy increases by a factor of 8). Also, the logarithm allows further cepstral mean subtraction, which is a channel normalisation technique.

Step 5: *Take the DCT of the log filterbank energies*. This action aims to decorrelates the filterbank energies (because of overlap between filterbanks). Also, as a result, "diagonal covariance matrices can be used to model the features in e.g a HMM classifier."

Step 6: *Keep DCT coefficients 2-13, discard the rest*. The reason is the higher DCT coefficients represent fast changes in the filterbank energies which actually degrade ASR performance. By dropping them, small improvement is achieved.

Function mfcc() in librosa package is used to extract MFCCs in this project. Besides MFCCs, wavelet features and spectral amplitudes are extracted and employed in kNN model to compare the results.

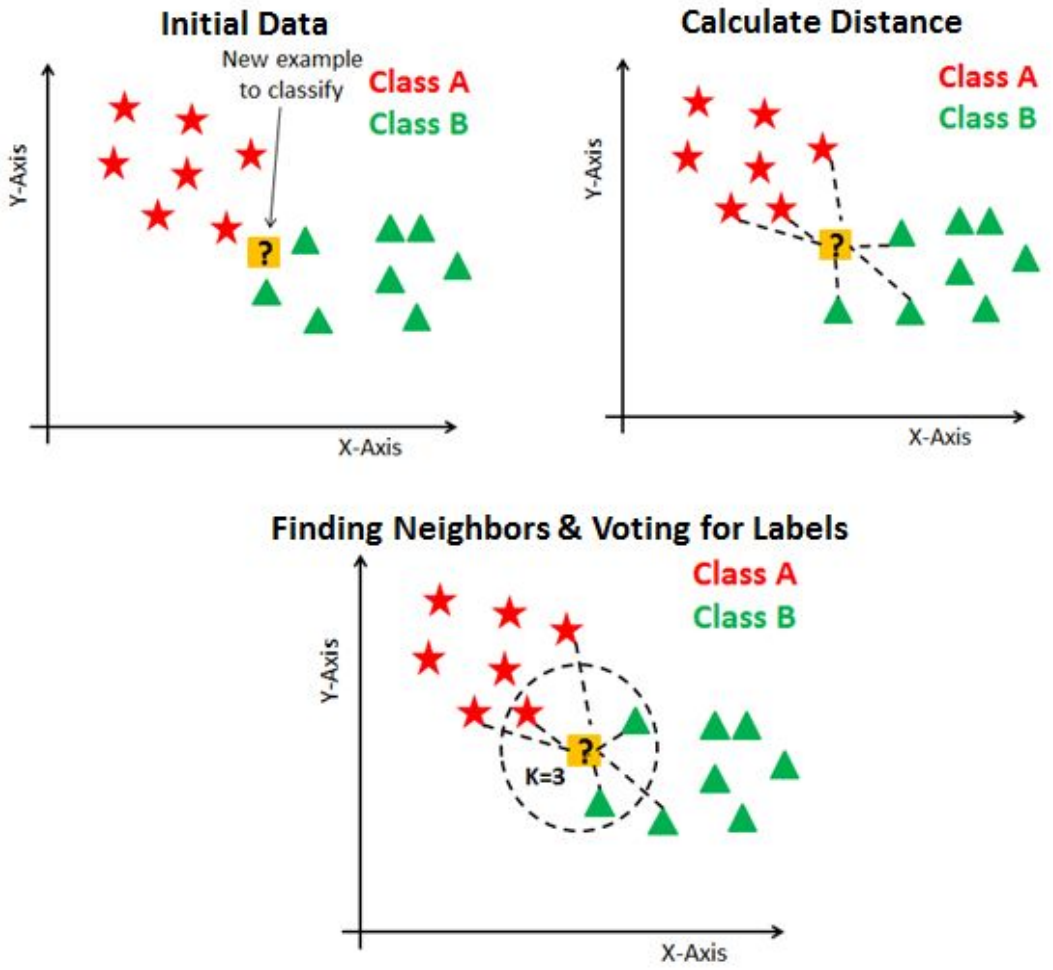
Classification

K-Nearest Neighbors (kNN) and Convolutional Neural Network (ConvNet/CNN) are employed to classify heart sounds.

- **k-Nearest Neighbors (kNN)**

The kNN algorithm is one of the simplest learning algorithms. It is a type of instance-based, competitive learning and lazy learning algorithms. It uses all of the data for training and doesn't assume anything about the underlying data. This feature is useful since there is no theoretical assumption such as linear-separability needed.

The intuition behind the kNN is simple. It calculates the distance of a new data point to all other training data points, and then selects the K-nearest data points, where K can be any integer. The data point is assigned to class to which the majority of the K data points belong.



Pre-implemented KNeighborsClassifier in sklearn package is used in this project. At first, 5 neighbors are used, then this number and other hyperparameters are optimized using grid search.

- **Convolutional Neural Network (ConvNet/CNN)**

In Deep Learning, a Convolutional Neural Network (ConvNet/CNN), which is inspired by the organization of animal Visual Cortex, is a regularized version of multilayer perceptrons (fully connected networks - each neuron in one layer is connected to all neurons in the next layer), most used in image analyzation. Since it can learn the filters, pre-processing required in a CNN is relatively lower as compared to other classification methods.

CNNs include an input, multiple hidden layers and an output. Typically, the hidden layers of a CNN consist of "a series of convolutional layers that convolve with a multiplication or other dot product". A RELU layer is commonly used as activation function, followed by additional convolutions such as pooling layers, fully connected layers and normalization layers.

- *Convolutional layer - the Kernel*

Each convolutional layer should have the following attributes:

Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth)

Convolutional kernels (simple as Kernel/Filter, K) whose width and height are hyperparameters, and whose depth must be equal to image depth; convolve the input and pass its result to the next layer.

By applying valid padding, convolved feature is reduced in dimensionality. Otherwise, the dimensionality is either increased or remains the same if same padding is applied.

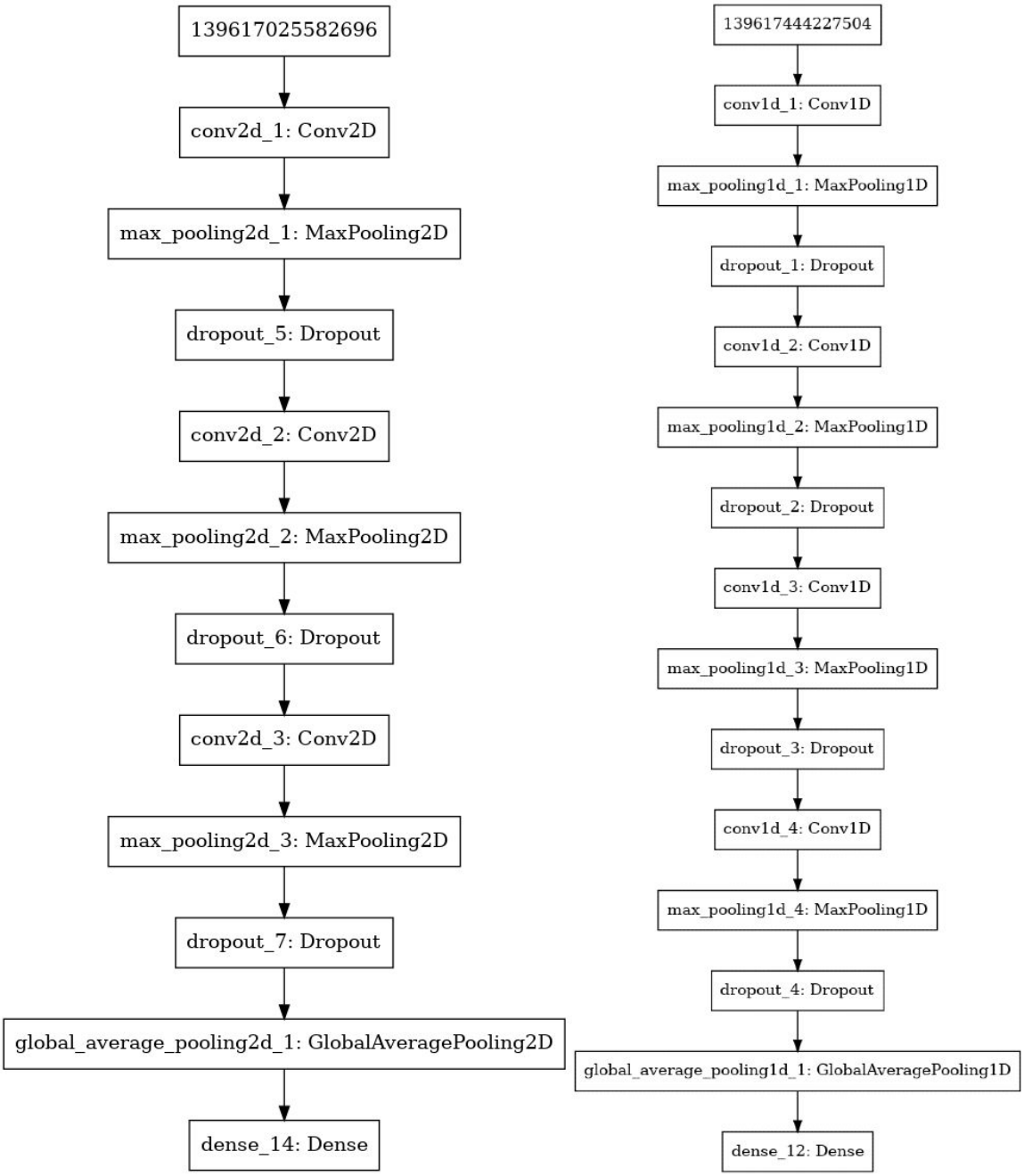
- *Pooling layer*

Similar to convolutional layers, pooling layers reduce the dimensions of data. Max pooling and average pooling are two types of pooling. As their self-explained names, max pooling returns the maximum value from the portion of the image covered by the Kernel, while average pooling returns its average. Since max pooling also performs as a Noise Suppressant, it is better than average pooling.

- *Fully Connected Layer*

"Fully connected layers connect every neuron in one layer to every neuron in another layer… The flattened matrix goes through a fully connected layer to classify the images".

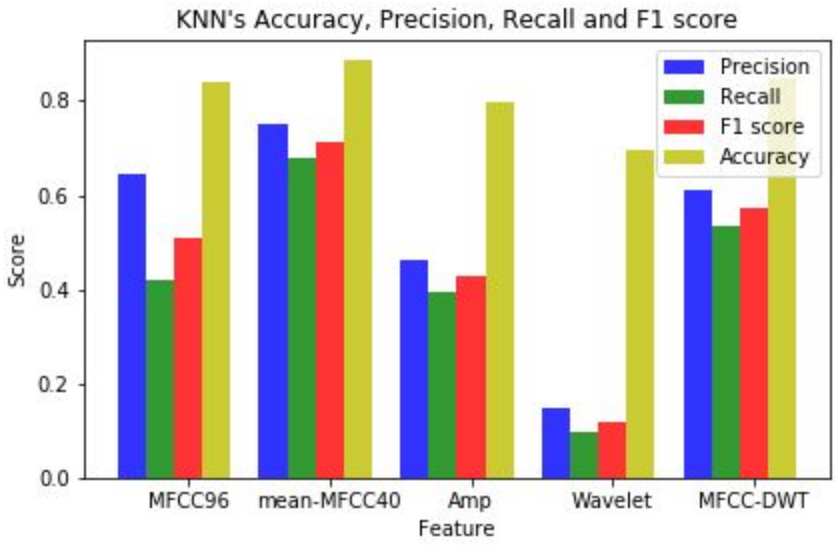
In this project, both 1D CNN and 2D CNN are built using Keras package in Python 3. Their detailed structures are shown in below picture.


139617025582696
conv2d_1: Conv2D
max_pooling2d_1: MaxPooling2D
dropout_5: Dropout
conv2d_2: Conv2D
max_pooling2d_2: MaxPooling2D
dropout_6: Dropout
conv2d_3: Conv2D
max_pooling2d_3: MaxPooling2D
dropout_7: Dropout
global_average_pooling2d_1: GlobalAveragePooling2D
dense_14: Dense
139617444227504
conv1d_1: Conv1D
max_pooling1d_1: MaxPooling1D
dropout_1: Dropout
conv1d_2: Conv1D
max_pooling1d_2: MaxPooling1D
dropout_2: Dropout
conv1d_3: Conv1D
max_pooling1d_3: MaxPooling1D
dropout_3: Dropout
conv1d_4: Conv1D
max_pooling1d_4: MaxPooling1D
dropout_4: Dropout
global_average_pooling1d_1: GlobalAveragePooling1D
dense_12: Dense

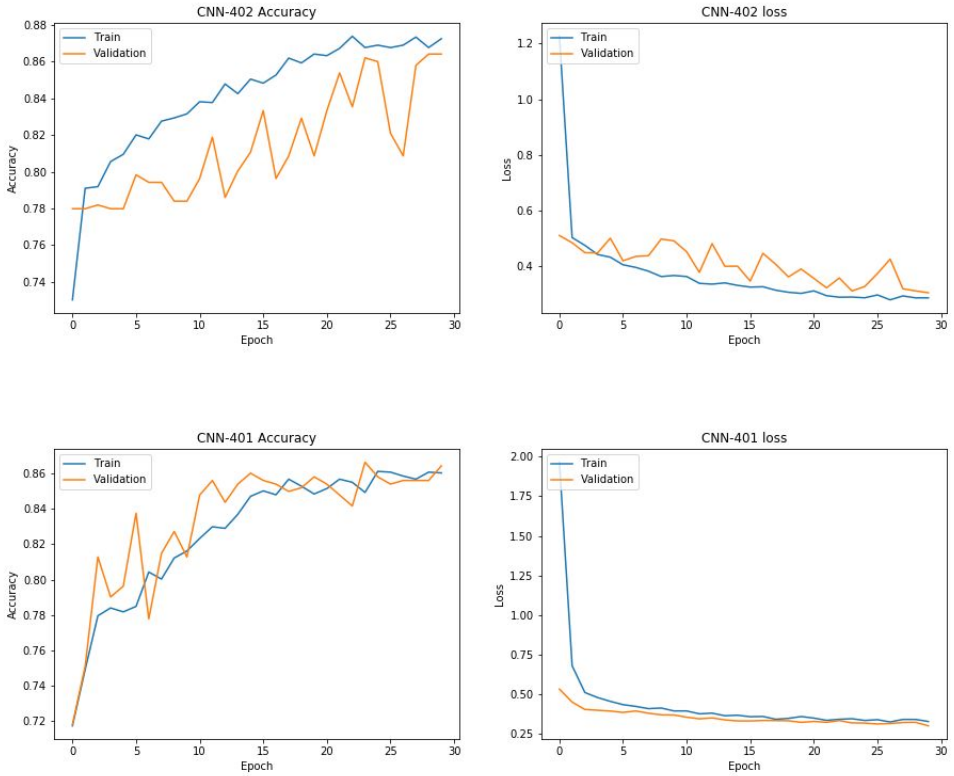
Analysis and Discussion

Model with highest performance				
Model	Accuracy	Precision	Recall	F score
kNN (40 mean-MFCCs)	0.89	0.76	0.71	0.74
CNN (2D, 40-MFCCs)	0.89	0.86	0.59	0.70

Among different features (flatten of 90-MFCCs, 40 mean-MFCCs, spectral amplitudes, wavelet-based feature, and combination of MFCCs and DWT) used for KNN model, the one using only 40 means of MFCCs has the highest performance with accuracy of 0.89, precision of 0.76, recall of 0.71 and F score of 0.74 (after optimization).



CNN models' performance is slightly lower than of kNN. Even though both two CNN models (2D and 1D) have high precision score (0.86 and 0.79), their recall scores are low (0.59 and 0.45). The 2D CNN model performs better than 1D CNN with accuracy of 0.89, and F score of 0.70 on test set. Below pictures virtualize their training process.



Conclusion and Future Direction

KNN and CNN models are trained and tested using different features, namely MFCCs, spectral amplitudes and wavelet features extracted from 3,240 .wav files (665 abnormal, 2575 normal). With F score of 0.74 and accuracy of 0.89, kNN model which uses only 40 mean-MFCCs conducts the leading result. CNN model's performance is slightly lower with F score of 0.70 and accuracy of 0.89. In the future, more consideration on choosing features and deciding some hyper-parameters will be taken to improve models.