Voice Pathology Detection Using Deep Learning

A Design Lab Report

Submitted by

Prajjwal Kumar –1901EE77

Department of Electrical and Electronics Engineering IIT Patna.

Abstract

This report describes a preliminary investigation of Voice Pathology Detection using Deep Neural Networks (DNN). I have used voice recordings of sustained sentence "*Guten Morgen, wie geht es Ihnen?*" (Good morning how are you) produced at normal pitch from German corpus Saarbruecken Voice Database (SVD). This corpus contains voice recordings and electroglottograph signals of more than 2000 speakers. The Idea behind this project is to Use Deep Neural Network on the feature extracted by Mel spectrogram generated on the raw speech signal to detect that the speaker's voice is pathological or non-Pathological. The Trained model achieved 87.16% accuracy on the data of 726 Female pathological voice and 382 Female Healthy voice.

Introduction

Voice quality issues could be a sign of a condition involving laryngeal problems. The auditory-perceptual evaluation is the most used method for assessing voice quality in clinical practice. Auditory-perceptual examination is a standardized process for determining whether or not a person's voice is abnormal. The clinician's direct audition is used to perform a subjective voice evaluation in this method. Because of the differences in listeners' perceptions of voice quality, auditory-perceptual evaluation of voice quality is not efficient. Laryngoscopic techniques such as indirect laryngoscopy, direct laryngoscopy, and telescopic video laryngoscopy, on the other hand, are invasive methods for viewing the vocal folds. Many laryngeal problems can be diagnosed using these procedures, which are routinely used for monitoring the larynx. These monitoring techniques, on the other hand, may cause discomfort to the patient and can be costly.

Many researchers are working on the method to differentiate between the Pathological and Non-Pathological voice quality.

In automatic classification of healthy and impaired voices, the feature is usually extracted from time, frequency and loudness using the Mel spectrogram.

Here in this project, I have used the Mel Spectrogram for the feature extraction of the Pathological and Non-Pathological voice and Implemented the Deep Neural Network on the same.

The report contains the Methodology, Applied Algorithm, Results and conclusion along with References.

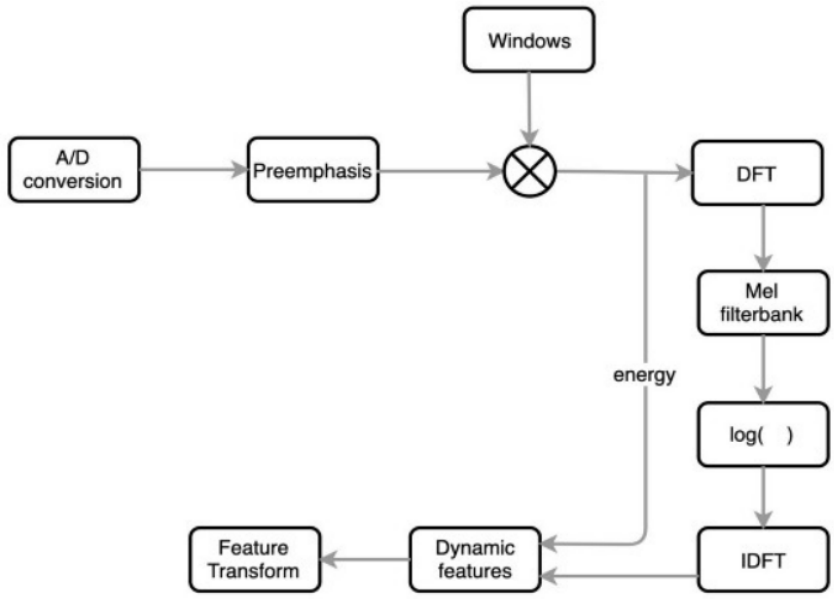
Methodology

The Saarbruecken Voice Database, which contains voice recordings and EGG signals from over 2 000 people, was used. It includes recordings of 687 healthy people (428 females and 259 males) and 1356 patients (727 females and 629 males) suffering from one or more of the 71 diseases. The data contains the recorded speech Sentence “Guten Morgen, wiegeht es Ihnen?” (“Good morning, how are you?”).

All the samples are sampled at 50KHz with 16-bit Resolution. From these set of speech data, I have extracted the Features using the **Mel Spectrogram.**

Windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by using the inverse DCT are all part of the Mel spectrogram feature extraction technique. Below is a full discussion of the many procedures involved in the Mel spectrogram feature extraction.

Block Diagram of Mel spectrogram generation:



Steps Involved in Mel spectrogram feature Extraction: -

1.A/D convertor:

We convert the analog signal into respective digital signal with a sampling frequency int the range of 8KHz to 16KHz.

2. Pre-emphasis: Filtering that emphasizes higher frequencies is referred to as pre-emphasis. Its goal is to balance the frequency range of vocal sounds with a sharp high-frequency roll-off. The glottal source has a slope of about -12 dB/octave for spoken sounds. When acoustic energy is radiated from the lips, however, the spectrum is boosted by about +6 dB/octave.

As a result, compared to the genuine spectrum of the vocal tract, a speech signal recorded with a microphone at a distance has a -6 dB/octave downward slope. Pre-emphasis thereby eliminates some glottal effects from the vocal tract characteristics. The following transfer function is the most widely used pre-emphasis filter.

$$H(z) = 1 - bz^{-1}$$

where the value of b controls the slope of the filter and is usually between 0.4 and 1.0.

3. Windowing and frame blocking: The voice signal is a slowly changing or quasi-stationary signal. Speech must be analyzed for a sufficiently short period of time in order to have stable acoustic features. As a result, speech analysis should always be performed on short segments when the speech signal is presumed to be steady. Short-term spectral measurements are commonly performed in 20-ms frames, with 10-ms intervals. The temporal properties of individual speech sounds can be followed by advancing the time window every 10 ms, and the 20 ms analysis window is usually sufficient to offer acceptable spectral resolution while also being short enough to detect major temporal aspects. The purpose of the overlapping analysis is that each speech sound of the input sequence would be approximately centered at some frame.

On each frame, a window is applied to taper the signal towards the frame boundaries. Generally, Hanning or Hamming windows are used. **Hamming window** and is of the form:

$$H(\theta) = 0.54 + 0.46 \cos\left[\left(\frac{2\pi}{N}\right) n\right]$$

The time and spectral responses of this filter are shown

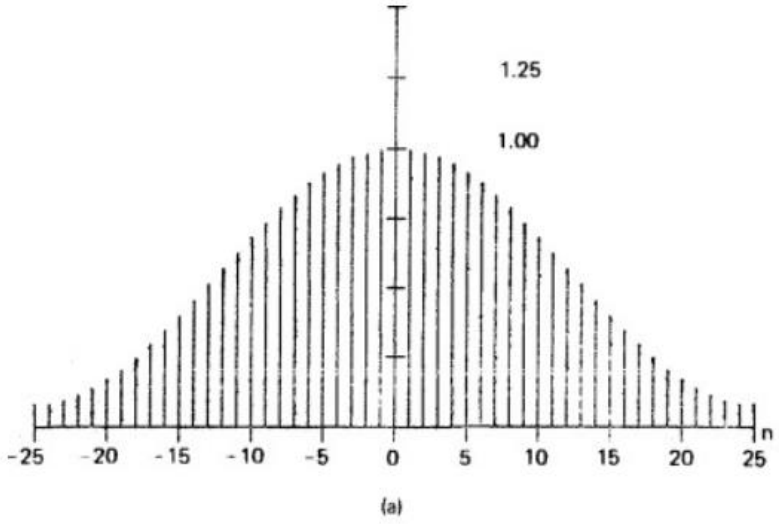


Fig – hamming window which is used in MFCC generation

highest sidelobe is attenuated with respect to the main lobe by 43 dB, and that the asymptotic rate of attenuation is 6 dB/octave.

This is done to enhance the harmonics, smooth the edges, and to reduce the edge effect while taking the **DFT** on the signal.

4. DFT spectrum: Each windowed frame is converted into magnitude spectrum by applying DFT.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}}; \qquad 0 \leq k \leq N-1$$

where N is the number of points used in the DFT calculation.

5. Mel spectrum: Mel spectrum is calculated by sending a Fourier converted signal through a Mel-filter bank of band-pass filters. A Mel is a unit of measurement based on the frequency perceived by the human ear. Because the human auditory system does not appear to perceive

pitch in a linear manner, it does not correspond linearly to the actual frequency of the tone. The Mel scale has a logarithmic frequency spacing below 1 kHz and a linear frequency spacing above 1 kHz.

The approximation of Mel from physical frequency can be expressed as:

$$f_{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

where **fMel** signifies the perceived frequency and **f** denotes the physical frequency in **Hz.**

Filter banks can be used in the time domain as well as the frequency domain. Filter banks are typically built in the frequency domain for MFCC computation.

On the frequency axis, the center frequencies of the filters are generally uniformly spaced. The warped axis, corresponding to the nonlinear function given in Eq. (A.3), is constructed to replicate the human ears perception. The triangular filter shaper is the most prevalent, and the Hanning filter can also be found in some circumstances.

The Mel spectrum of the magnitude spectrum X(k) is computed by multiplying the magnitude spectrum by each of the of the triangular Mel weighting filters.

$$s(m) = \sum_{k=0}^{N-1} \left[|X(k)|^2 H_m(k)\right]; \qquad 0 \le m \le M-1$$

where M is total number of triangular Mel weighting filters.

Hm(k) is the weight given to the kth energy spectrum bin contributing to the mth output band and is expressed as

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{2(k-f(m-1))}{f(m)-f(m-1)}, & f(m-1) \le k \le f(m) \\ \frac{2(f(m+1)-k)}{f(m+1)-f(m)}, & f(m) < k \le f(m+1) \\ 0, & k > f(m+1) \end{cases}$$

with m ranging from 0 to M − 1.

This is the entire Process of Mel Spectrogram feature Extraction which I have done using the **Librosa library** in Python. The output of librosa Mel Spectrogram feature extraction I have stored in the 2-Dimensional array and further I have applied the DNN architecture to that feature Extracted data.

The Generated Mel spectrogram for one of the raw audio sample is given as

The part of the code through which I have got the amplitude, sampling rate and number of fft window is following:

```
from scipy.io import wavfile

def readAudio(audio):
    fs, amp = wavfile.read(audio)
    dt = 1/fs
    n = len(amp)
    t = dt*n

    if t > 1.0:
        amp = amp[int((t/2 - 0.5)/dt):int((t/2 + 0.5)/dt)]
        n = len(amp)
        t = dt*n

    return(amp, fs, n, t)
```

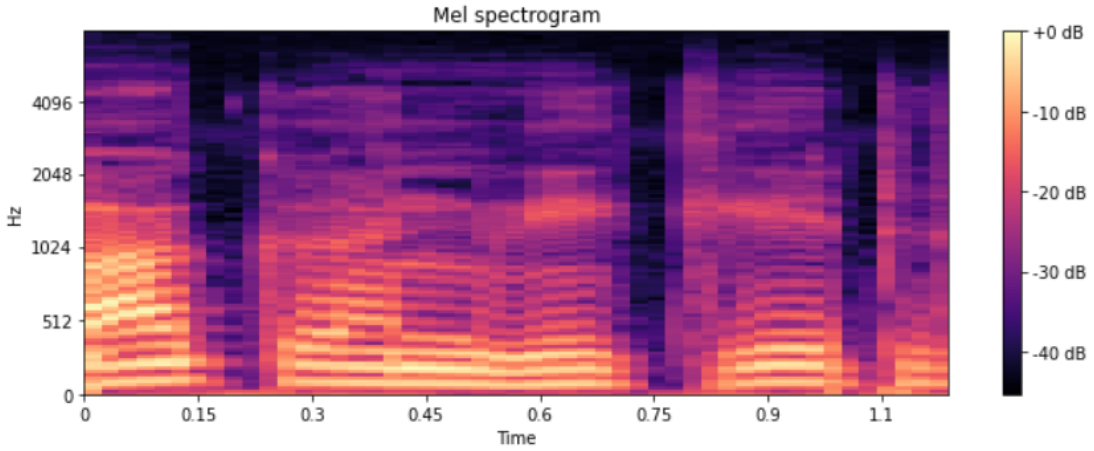
This function will return the amplitude, sampling rate and number of fft window which we will use further to generate Mel spectrogram.

```
amp, fs, n ,t = readAudio('/content/drive/MyDrive/828-phrase.wav')
```

Here we make call to the above function to get amplitude, and sampling rate respectively

```
S = librosa.feature.melspectrogram(y=amp*1.0, sr=fs, n_fft=frame_length, hop_length=overlap, power=1.0)
fig = plt.figure(figsize=(10,4))

librosa.display.specshow(librosa.power_to_db(S,ref=np.max), y_axis='mel', fmax=8000, x_axis='time')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel spectrogram')
plt.tight_layout()
```



Total 128 feature of this particular Mel Spectrogram I have got as output. This feature further used in DNN which is explained in the Algorithm section.

Total number of features for the mel spectrogram is 128 which can be seen in the output. And we will use this X 2d array as feature for applying the DNN.

```
print('X shape: ', X.shape, 'y shape: ', y.shape)
```

```
[[8.50187039e+01 2.31331329e+01 1.08422242e+01 ... 7.66225128e-02
  7.62469166e-02 7.96878106e-02]
 [8.50187039e+01 2.31331329e+01 1.08422242e+01 ... 7.66225128e-02
  7.62469166e-02 7.96878106e-02]
 [1.72901807e+01 5.52023835e+00 3.91344870e+00 ... 3.09657223e-02
  2.73865739e-02 2.88062488e-02]
 ...
 [6.59347986e+01 3.20557703e+01 1.68428226e+01 ... 7.02197978e-02
  6.96418193e-02 7.36294108e-02]
 [1.66434211e+01 5.59839348e+00 3.46945692e+00 ... 5.76294108e-02
  5.89561284e-02 6.12257844e-02]
 [1.66434211e+01 5.59839348e+00 3.46945692e+00 ... 5.76294108e-02
  5.89561284e-02 6.12257844e-02]]
X shape:  (2216, 128) y shape:  (2216,)
```

Algorithm Applied (After Feature Extraction)

1.The 2-D feature extracted array we got as output of the Mel Spectrogram extraction is firstly normalized.

2.spilt the data or divide the data into 80% training and 20% test model using the sklearn model library in python

3.Creating the Deep Neural Network model .I have used three-layer 1st layer is Dense layer having 100 neurons and activation function is “relu”. 2nd layer has 50 neurons and “relu” activation function and 3rd or final one has “softmax” as activation function.

```
model=Sequential()

###first layer
model.add(Dense(100,input_shape=(128,)))
model.add(Activation('relu'))

###second layer
model.add(Dense(50))
model.add(Activation('relu'))

###final layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 100)               12900

 activation_3 (Activation)   (None, 100)               0

 dense_4 (Dense)             (None, 50)                5050

 activation_4 (Activation)   (None, 50)                0

 dense_5 (Dense)             (None, 2)                 102

 activation_5 (Activation)   (None, 2)                 0

=================================================================
Total params: 18,052
Trainable params: 18,052
Non-trainable params: 0
_________________________________________________________________
```

Fig – summary of the given model

4.compile the model with loss function 'categorical_crossentropy' and "adam" as optimizer.

5. Model Training using 100 epochs and 32 batch size at a time.

Code for the compiling and Training the data.

```
model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
```

```
## Trianing my model
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath='Desktop/sbvoicedb_RdLu8Y.hdf5',
                               verbose=1, save_best_only=True)
start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_split=0.10, callbacks=[checkpointer], verbose=1)


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

```
Epoch 1/100
34/50 [===================>.....] - ETA: 0s - loss: 470.9094 - accuracy: 0.6351
Epoch 1: val_loss improved from inf to 248.62569, saving model to Desktop\sbvoicedb_RdLu8Y.hdf5
50/50 [==============================] - 0s 4ms/step - loss: 396.5645 - accuracy: 0.6455 - val_loss: 248.6257 - val_accuracy: 0.6854
Epoch 2/100
36/50 [====================>.....] - ETA: 0s - loss: 263.5741 - accuracy: 0.6293
Epoch 2: val_loss improved from 248.62569 to 152.25256, saving model to Desktop\sbvoicedb_RdLu8Y.hdf5
50/50 [==============================] - 0s 2ms/step - loss: 241.7011 - accuracy: 0.6631 - val_loss: 152.2526 - val_accuracy: 0.7416
Epoch 3/100
42/50 [========================>.....] - ETA: 0s - loss: 113.8244 - accuracy: 0.7329
Epoch 3: val_loss improved from 152.25256 to 86.66239, saving model to Desktop\sbvoicedb_RdLu8Y.hdf5
50/50 [==============================] - 0s 2ms/step - loss: 106.9144 - accuracy: 0.7415 - val_loss: 86.6624 - val_accuracy: 0.7191
Epoch 4/100
```

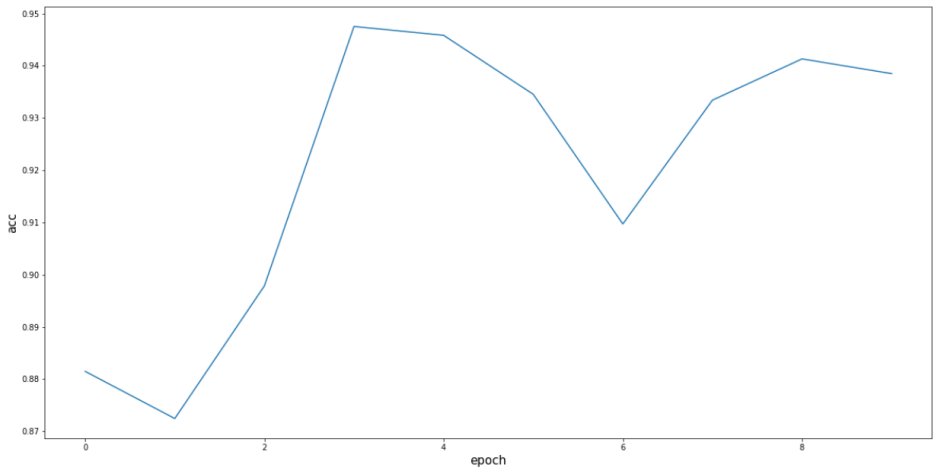
[Code for the above Report]

Results:

Accuracy = 87.16%.

```
test_accuracy=model.evaluate(X_test,y_test,verbose=0)
print(test_accuracy[1])
```

```
0.8716216087341309
```



Conclusion:

This project basically aims on the fact that Mel spectrogram is one of the best ways to visualize the audio signals. Speech Recognition is a supervised learning task. In the speech recognition problem input will be the audio signal and we have to predict the text from the audio signal. We can't take the raw audio signal as input to our model because there will be a lot of noise in the audio signal. MFCC is the widely used technique for extracting the features from the audio signal. Since here we have got good accuracy of around 87% hence using the MFCC spectrogram is beneficial and we can also use CNN and LSTM models on this to improve the accuracy as well as Precision. These techniques, which are commonly used for monitoring the larynx, make the diagnosis of many laryngeal disorders possible without causing the discomfort to the patients.

References:

1. Research Article Voice Disorder Classification Based on Multitaper Mel Frequency Cepstral Coefficients Features – Hindawi

2. Voice Pathology Detection Using Deep Learning: a Preliminary Study
3. MFCC Features article from springer

Submitted to: communication panel – Dr.Preetam Kumar (**Supervisor**) , Dr.Sumanta Gupta, Dr. Sudhir Kumar.