Cell [1]

This code is importing the necessary packages for a machine learning project that involves audio processing. Specifically, it is importing packages for reading and manipulating audio files, as well as building and training deep learning models. It imports necessary libraries such as librosa, numpy, pandas, tensorflow, and sklearn.

Cell [3] - Cell [6]

In this code block, we set mypath to the directory path where the audio files are located. We then use the listdir function from the os module to get a list of all files in the directory, and filter the list to only include files that end with the .wav extension. We store the resulting list of file names in the filenames variable.

Then it reads the filenames of the audio files and extracts the patient IDs from them. The extract\_features function takes the file name as an input and returns the MFCC (Mel-frequency cepstral coefficients) of the audio.It uses the librosa library to load and extract the MFCC features. The MFCC features are then padded with zeros to make the length of all MFCC equal to the max\_pad\_len, which is set to 862. If an error is encountered while parsing the file, the function returns None.The code also reads the diagnosis information of each patient from the CSV file patient\_diagnosis.csv.

Cell [7] - Cell [9]

filepaths is a list of full paths to the audio files that were extracted in filenames.

This line of code creates a list of file paths by joining the directory path mypath and the file names in the list filenames.

The join method is used to concatenate the two strings, which creates the full file path for each file in the directory. The resulting list filepaths contains the full paths of all the audio files in the directory.

p\_diag is a pandas dataframe that contains the patient ID and diagnosis information. The patient IDs of the audio files are matched with the patient IDs in p\_diag, and the corresponding diagnoses are extracted and stored in the labels array.

Cell [10] - Cell [11]

In this code block, the features of each audio file are extracted by calling the extract\_features() function, and appending the extracted features to the features list. Finally, the features list is converted to a NumPy array for further processing

The plt.figure(figsize=(10, 4)) command sets the figure size to be 10 inches in width and 4 inches in height. librosa.display.specshow plots the MFCC matrix with time on the x-axis and the frequency on the y-axis. plt.colorbar() adds a colorbar to the plot. plt.title('MFCC') sets the title of the plot to be "MFCC". plt.tight\_layout() is used to adjust the padding between the plot and the edge of the figure. Finally, plt.show() displays the plot.

Cell [12] - Cell[14]

This command converts the features list into a numpy array. The features list contains the extracted MFCC features from the audio files.This code deletes the audio files from the dataset that have a rare diagnosis, in this case, 'Asthma' and 'LRTI'. It then prints the class counts for the remaining diagnoses.

Cell [15]

This code plots a bar chart showing the count of each disease in the sound files. The x-axis represents the different diseases and the y-axis represents the count of sound files associated with each disease. The height of each bar represents the count of sound files associated with the corresponding disease. The y\_pos variable is used to set the positions of the ticks on the x-axis. The plt.bar() function is used to plot the bar chart, and the plt.xticks() function is used to label the ticks on the x-axis. The plt.title(), plt.xlabel(), and plt.ylabel() functions are used to set the title, x-axis label, and y-axis label of the chart, respectively.

Cell [16] - Cell [18]

The code is preparing the data for training the model.

* le = LabelEncoder(): creates a LabelEncoder object used to transform categorical labels into numerical values that can be used in a machine learning model.
* i\_labels = le.fit\_transform(labels1) applies the label encoder to the labels1 variable, which likely contains a list of categorical labels. The fit\_transform() method of the LabelEncoder class fits the encoder to the label data and then transforms the labels into numerical values. The transformed labels are stored in the i\_labels variable.
* oh\_labels = to\_categorical(i\_labels) converts the numerical labels stored in i\_labels into one-hot encoded vectors using the to\_categorical() function from the Keras library. One-hot encoding is a common technique for representing categorical data in a format that can be used in machine learning models. It converts each categorical label into a binary vector of length equal to the number of categories, with a value of 1 in the position corresponding to the category and 0s elsewhere. The resulting one-hot encoded labels are stored in the oh\_labels variable.
* features1 = np.reshape(features1, (\*features1.shape,1)): adds a channel dimension to the feature data for compatibility with the convolutional neural network.
* x\_train, x\_test, y\_train, y\_test = train\_test\_split(features1, oh\_labels, stratify=oh\_labels, test\_size=0.2, random\_state = 42): splits the data into training and test sets with a 80:20 ratio. The stratify parameter ensures that the ratio of labels is maintained in both the training and test sets.

Cell [19]

This code constructs a Convolutional Neural Network (CNN) model using the Keras Sequential API. The model has four convolutional layers with 16, 32, 64, and 128 filters respectively, each followed by a max pooling layer with pool size of 2 and a dropout layer with a dropout rate of 0.2 to prevent overfitting. The model ends with a global average pooling layer and a dense output layer with a softmax activation function, which outputs class probabilities. The input shape of the model is (num\_rows, num\_columns, num\_channels), where num\_rows and num\_columns are the dimensions of the input MFCC features and num\_channels is set to 1 since we only have one audio channel.

Filter : a set of weights that are learned during the training process to detect specific features or patterns in the input data.

Pooling : technique used in neural networks to reduce the spatial dimensions of the input data. It works by dividing the input data into a set of non-overlapping regions and then computing the maximum, average, or other function of the values within each region. This reduces the size of the data and helps prevent overfitting by extracting the most important features from the input data.

Overfitting : a problem that occurs in machine learning when a model is trained too well on the training data, so that it begins to fit the noise in the data instead of the underlying patterns. As a result, the model does not generalize well to new data that it has not seen before. The model is said to have overfit the training data and has poor performance on the test data.

Cell [20] - Cell [21]

The first line compiles the model using categorical cross-entropy loss as the loss function, using the Adam optimizer to update the model weights, and keeping track of the accuracy metric during training.

The model.summary() function is used to display the architecture of the model, including the output shape of each layer and the total number of parameters in the model.

The last three lines are used to calculate the pre-training accuracy of the model on the test set, which is the accuracy of the model before any training is performed.

The model.evaluate method is used to evaluate the trained model on a test dataset. It takes in the test data (x\_test) and their corresponding labels (y\_test), and returns the loss value and the accuracy score of the model on the test data. The verbose argument controls whether to show the progress bar or not. A value of 1 shows the progress bar during evaluation, while a value of 0 does not.The accuracy is then printed .

Cell [23]

The code is training the constructed model on the training data x\_train and y\_train, with a batch size of 128 and for 250 epochs. The validation data x\_test and y\_test is used to evaluate the performance of the model during training. Additionally, the code saves the best performing model (based on the validation accuracy) in the file mymodel2\_{epoch:02d}.h5. The training progress is displayed during training, and the total time taken for training is printed at the end. The ModelCheckpoint callback used here allows us to save the model at intervals during training, so we can select the best model and avoid overfitting.

Epochs : a complete iteration through a dataset during training which includes a single pass through the entire training dataset, followed by testing the model on the testing dataset.

Generally, increasing the number of epochs can improve the performance of the model, up to a certain point where the model starts to overfit the training data.

Callbacks : Callbacks in Keras are functions that can be applied at certain stages of the training process, such as at the end of each epoch, during early stopping, or when a performance threshold has been reached. In this specific case, the ModelCheckpoint callback is used to save the model after every epoch, only if the val\_accuracy has improved. This ensures that the best version of the model is saved during the training process, rather than only at the end of the training.

Cell [24] - Cell [25]

It is evaluating the model on the training and testing data sets, and printing the resulting accuracy for each.In the code , we used the trained model to make predictions on the test dataset. The model.predict() method returns the label scores for each instance in the test dataset. We then used np.argmax() method to get the index of the maximum score for each instance which represents the predicted class. Similarly, we used np.argmax() method on the y\_test array to get the true classes.

Finally, we assigned the number of classes to n\_classes which is used in calculating the classification report and confusion matrix in the following code.

Cell [26] - Cell [28]

First it is computing the False Positive Rate (FPR), True Positive Rate (TPR) and the Area Under the ROC Curve (AUC) for each class. The FPR and TPR are then saved into dictionaries indexed by the class number.

c\_names is a list that contains the names of the classes in the order in which they appear.

classification\_report is a function from the sklearn.metrics module that is used to print the classification report of a classification model. The function takes three arguments:

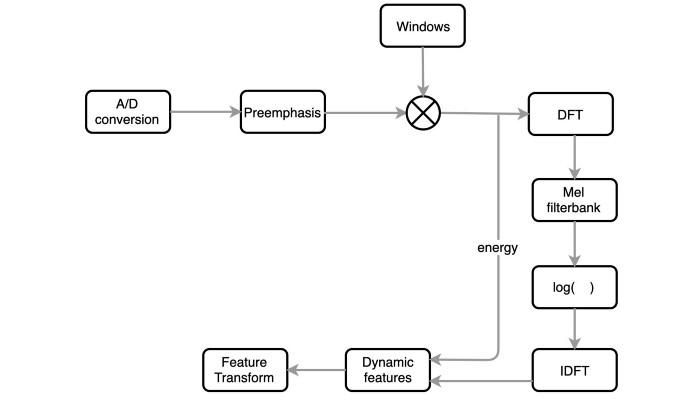
* y\_true: The true labels of the test set.
* y\_pred: The predicted labels of the test set.
* target\_names: A list of class names that will be used to label the rows in the report.

The report includes various performance metrics for each class such as precision, recall, F1-score, and support. The report also includes an overall performance summary.

MFCC:-

MFCC is the widely used technique for extracting the features from the audio signal.

The road map of the MFCC technique is given below.



### A/D Conversion:

In this step, we will convert our audio signal from analog to digital format with a sampling frequency of 8kHz or 16kHz.

### Preemphasis:

Preemphasis increases the magnitude of energy in the higher frequency. Boosting the energy in higher frequencies will improve the phone detection accuracy thereby improving the performance of the model.

Preemphasis is done by the first order high-pass filter.

### Windowing:

The MFCC technique aims to develop the features from the audio signal which can be used for detecting the phones in the speech. But in the given audio signal there will be many phones, so we will break the audio signal into different segments with each segment having 25ms width and with the signal at 10ms apart .On average a person speaks three words per second with 4 phones and each phone will have three states resulting in 36 states per second or 28ms per state which is close to our 25ms window.

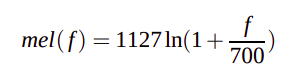
### DFT( Discrete Fourier Transform):

We will convert the signal from the time domain to the frequency domain by applying the dft transform. For audio signals, analyzing in the frequency domain is easier than in the time domain.

### Mel-Filter Bank:

The way our ears will perceive the sound is different from how the machines will perceive the sound. Our ears have higher resolution at a lower frequency than at a higher frequency. So if we hear sound at 200 Hz and 300 Hz we can differentiate it easily when compared to the sounds at 1500 Hz and 1600 Hz even though both had a difference of 100 Hz between them. Whereas for the machine the resolution is the same at all the frequencies. It is noticed that modeling the human hearing property at the feature extraction stage will improve the performance of the model.

So we will use the mel scale to map the actual frequency to the frequency that human beings will perceive. The formula for the mapping is given below.



### Applying Log:

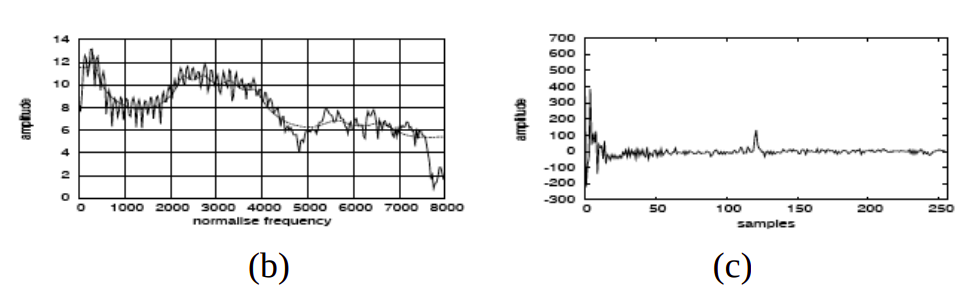
Humans are less sensitive to change in audio signal energy at higher energy compared to lower energy. Log function also has a similar property, at a low value of input x gradient of log function will be higher but at high value of input gradient value is less. So we apply log to the output of Mel-filter to mimic the human hearing system.

### IDFT:

In this step, we are doing the inverse transform of the output from the previous step.Note that the periods in the time domain and frequency domain are inverted after the transformations. So, the frequency domain’s fundamental frequency with the lowest frequency will have the highest frequency in the time domain.

Note: The inverse of the log of the magnitude of the signal is called a cepstrum.

The below figure shows the signal sample before and after the idft operation.



The peak frequency at the rightmost in figure(c) is the fundamental frequency and it will provide information about the pitch and frequencies at the rightmost will provide information about the phones. We will discard the fundamental frequency as it is not providing any information about phones.

The MFCC model takes the first 12 coefficients of the signal after applying the idft operations. Along with the 12 coefficients, it will take the energy of the signal sample as the feature. It will help in identifying the phones. The formula for the energy of the sample is given below.

### 

### Dynamic Features:

Along with these 13 features, the MFCC technique will consider the first order derivative and second order derivatives of the features which constitute another 26 features.

Derivatives are calculated by taking the difference of these coefficients between the samples of the audio signal and it will help in understanding how the transition is occurring.

So overall MFCC technique will generate 39 features from each audio signal sample which are used as input for the speech recognition model.