

EARLY DIAGNOSIS OF COVID-19 USING SOUND SIGNAL PROCESSING

J Component Project Report for the course

ECE2006 – DIGITAL SIGNAL PROCESSING

by

**PEMMAREDDY SREEVARDHAN-20BEC1251
N G JAGADEESWAR – 20BEC1345
M V SAI JASWANTH – 20BEC1250**

Submitted to

Dr. R. Ramesh



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF ELECTRONICS ENGINEERING
VELLORE INSTITUTE OF TECHNOLOGY
CHENNAI - 600127**

NOVEMBER 2022

Certificate

This is to certify that the Project work titled “**EARLY DIAGNOSIS OF COVID-19 USING SOUND SIGNAL PROCESSING**” is being submitted by **PEMMAREDDY SREEVARDHAN (20BEC1251), N G JAGADEESWAR (20BEC1345), M V SAI JASWANTH (20BEC1250)** for the course **Digital Signal Processing**, is a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University.

Dr. R. Ramesh
Associate Professor
School of Electronics Engineering (SENSE)
VIT University, Chennai

ACKNOWLEDGEMENT

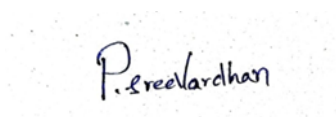
We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Ramesh**, Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Susan Elias**, Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the school towards our project and for her unstinting support

We express our thanks to our Head of the Department **Dr. Mohanaprasad. K** for his support throughout the course of this project.

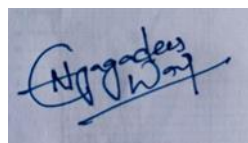
We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.



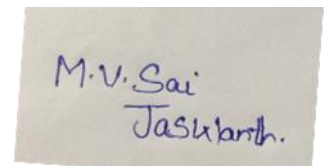
Signature

Name of the student
P SREEVARDHAN



Signature

Name of the student
N G JAGADEESWAR



Signature

Name of the student
M V SAI
JASWANTH

ABSTRACT

The Coronavirus has rapidly expanded around the world during the last two to three years. There is currently no known cause for this outbreak, however research is ongoing to find a cure for this illness. Due to time and money constraints, it is not viable to test for the coronavirus given the daily increase in cases. Machine learning has become increasingly dependable in the medical industry in recent years. Using machine learning to anticipate COVID-19 in patients will speed up the turnaround time for test results and direct medical personnel to treat patients as needed.

This thesis major objective is to create a machine learning model that can determine whether a patient has COVID-19. A literature review and experiment are planned to find a viable algorithm for such a model. In this model, we employed SVMs (Support Vector Machines) to make predictions. to evaluate the characteristics that affect the prediction model

Keywords: COVID-19, SVM.

Table of Contents

Chapter No.	Title	Page No.
	Acknowledgement	iii
	Abstract	iv
	List of Figures	vi
1	Introduction	07
	1.1 Objectives	07
	1.2 Scope	08
	1.3 Applications	08
2	Design/ Implementation	
	2.1 Design Approach	09
	2.2 Feature Extraction	10
	2.3 Trained Classifier	12
	2.4 Training Strategies	14
	2.5 Overview of software	17
	2.6 Summary	18
3	Result and Analysis/Testing	19
	3.1 JUPYTER Implementation	19
	3.2 JUPYTER Output Screenshots	25
	3.3 Summary	30
4	Conclusion and Future Enhancement	31
5	References	32
6	Bio Data	33

List of Figures

Figure No.	Title	Page No.
2.1.2a	Block Diagram	15
2.1.4.a	Architecture	16
3.2.a	Features for each class	25
3.2.b	Showing dimensions of training and testing data	26
3.2.c	Training of data	27
3.2.d	Loss curves	27
3.2.e	Accuracy curves	28
3.2.f	Result for Negative audio signal	28
3.2.g	Spectrogram for Negative audio signal	29
3.2.h	Result for Positive audio signal	29
3.2.i	Spectrogram for Positive audio signal	30

CHAPTER – 1

INTRODUCTION

The inability to conduct large-scale testing has turned into humanity's fatal flaw during this COVID19 pandemic. More than 30 medical disorders other than COVID-19 share symptoms with COVID-19. Therefore, even if a person exhibits COVID-19 symptoms, it is still possible that they are caused by a different, non-COVID-19 medical issue. At this point, COVID-19 testing is both important and dangerous for the person. The in-person testing method poses a major threat to both the patient and the medical staff since there is a higher risk of infection in the hospital or the location of the in-person testing because those are the gathering places for both people with and without COVID-19.

1.1 Objective

A perilous and disastrous tragedy for humanity has resulted from the COVID-19 pandemic brought on by the SARS-CoV-2 virus. The Reverse-Transcription Polymer Chain Reaction (RT-PCR) test is now used to diagnose COVID-19. However, in places where resources are scarce, this method is costly, time-consuming, and difficult to get. To get over these restrictions, an interpretable and COVID-19 diagnosis AI framework is created based on the cough sounds characteristics and symptoms metadata. The demographic, symptom, and cough features are included into an AI-based diagnostic tool for COVID-19 screening in this work, which also achieves mean accuracy, precision, and precision in the Above mentioned tasks.

1.2 Scope

The creation of a machine learning model for patients' COVID-19 is the main goal of this study. We also try to pinpoint the characteristics of patient clinical data that can affect COVID-19's predictive performance. The focus of this study is not on external variables like the climate or any other environmental factors that could affect the findings.

1.3 Applications

- Audio-Only Cough Monitors
- Cough Monitors with Mixed Signals
- The Ideal Cough Frequency Monitors
- characteristics of the diseases and their symptoms.

CHAPTER – 2

DESIGN / IMPLEMENTATION

2.1 Design Approach

We have created an end-to-end ML-based framework that can incorporate cough samples and directly predict binary classification labels, signalling the feasibility of COVID-19. This study was motivated by the present advancement of ML-based audio applications. We use audio characteristics such as Mel-Frequency Cepstral Coefficients, Mel-Scaled Spectrogram, Tonal Centroid, Contrast as the foundation of our suggested technique, followed by feature fusion. The output of the feature fusion is passed on to the trained classifier layer, which is made up of 2 classification techniques: Support Vector Machine (SVM), Random Forest (RF).

In addition, to select an optimized COVID-19 cough diagnosis model, we use the MCDM method that considers the decision matrix generated from different evaluation criteria outlined. After that, we calculate the relative closeness score of each training strategy by integrating entropy. Then, we use two ensemble strategies (soft ensemble and hard ensemble) to rank the models. We further analyze the effect of feature dimensionality reduction. Finally, we fed the selected features into the best classifier to detect COVID-19. The following sections outline the dataset description, the proposed method (including feature extraction and classification), the training strategies used, and the details of the optimization techniques used to select the best model.

2.2 Feature extraction method

As 44.1 kHz is a common frequency for audio applications, it is used to sample the sound waveform taken into account throughout the feature extraction process to maintain uniformity. The librosa module for Python is used to extract five spectral properties from the sampled audio, including Mel-Frequency Cepstral Coefficients, Mel-Scaled Spectrogram, Tonal Centroid, Chromagram, and Spectral Contrast.

2.2.1 Mel-Frequency Cepstral Coefficients (MFCCs):

MFCCs have already shown their usefulness through the analysis of dry and wet cough detection as well as highlighted as successful features for audio analysis. In the feature extraction of MFCC, after the windowing operation, fast fourier transform (FFT) applies to find the power spectrum of each frame. Afterward, the Mel scale is used to perform filter bank processing on the power spectrum. Mel-scaled filters are calculated from physical frequency (f) by the following Equation. After converting the power spectrum to the logarithmic domain, discrete cosine transform (DCT) is applied to the audio signal to measure the MFCC coefficients.

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

2.2.2 Mel-Scaled Spectrogram:

In ML applications concerning audio analysis, we often need to represent the power spectrogram in the Mel scale domain. The feature extraction process of the Mel-scaled Spectrogram includes several steps to generate the spectrogram. Before calculating the FFT, we set the window size to 2048 and the hop length to 512. After that, we set the number of Mels to 128, which is the evenly spaced frequency. Finally, the magnitude of the signal is decomposed into components corresponding to the frequencies in the Mel scale.

2.2.3 Tonal Centroid:

The tonal centroid feature is a way of projecting a 12-bin tuned chromagram onto a 6-dimensional vector, as described in Equation

$$\zeta_n(d) = \frac{1}{\|c_n\|_1} \sum_{l=0}^{11} \Phi(d, l) c_n(l), \quad 0 \leq d < 5; \quad 0 \leq l \leq 11$$

where ζ_n is the tone centroid vector, and for the time frame n is given by the product of the transformation matrix, Φ , and the chroma vector c .

2.2.4 Chromagram:

We calculate the chromatogram from the short-time Fourier transform (STFT) power spectrum. We initialize the window size to 2048 and the hop length to 512. The number of chroma bins generated is 12. Finally, it extracts the normalized energy of each chroma bin on each frame, which is the required feature vector.

2.2.5 Spectral Contrast:

First, perform FFT on the audio samples to obtain the frequency spectrum. Using several Octave-scale filters, the frequency domain is partitioned into sub-bands. In the feature extraction process, the number of frequency bands is set to be 6. The strength of spectral valleys, peaks, and their differences are evaluated in each sub-band, as stated in Equations. After being converted to the logarithmic domain, the original spectral contrast features will be mapped to the orthogonal space.

$$Peak_k = \log \left\{ \frac{1}{\alpha N} \sum_{i=1}^{\alpha N} x_{k,i} \right\}$$

$$Valley_k = \log \left\{ \frac{1}{\alpha N} \sum_{i=1}^{\alpha N} x_{k,N-i+1} \right\}$$

$$SC_k = Peak_k - Valley_k$$

2.3 Trained classifiers

In our proposed method for classification, we take into account ten machine learning (ML) algorithms, including Extremely Randomized Trees (Extra-Trees), Support Vector Machine (SVM), Random Forest (RF), Adaptive Boosting (AdaBoost), Multilayer Perceptron (MLP), Extreme Gradient Boosting (XGBoost), Gradient Boosting (GBoost), Logistic Regression (LR), k-Nearest Neighbor (k-NN) (HGBBoost). We'll give a quick overview of each of the several classifiers considered in our experimental evaluation in the sections that follow.

Extremely Randomized Trees (Extra-Trees) is a classifier that can fit multiple random decision trees to each sub-sample of the dataset, so it can control overfitting and uses the average to improve detection accuracy. The Extra-Trees classifier has proven useful in diagnosing patients with chronic obstructive pulmonary disease

Support Vector Machine (SVM) is a popular supervised technique that can effectively perform classification tasks. Several SVM kernels (such as Gaussian function, polynomial function, or quadratic function) can be used during the classification task. Some previous studies have successfully applied SVM to detect COVID-19 in audio samples.

Random Forest (RF) is a collection of decision trees widely used in classification tasks. By growing a combination of trees and voting for each category of trees, we can observe significant classification accuracy. Random Forest has achieved success in classifying cough, breath, and sound events [\[13\]](#).

Adaptive Boosting (AdaBoost) is a classifier that first fits the classifier to the original dataset, and then fits other copies of the classifier to the same dataset. However, the weights of misclassified instances are adapted to force successive classifiers to pay more attention to hard events.

Multilayer Perceptron (MLP) has adapted to the concept of human biological neural networks and can learn non-linear relationships. The training of the network depends on iteration, bias, weight adjustment, learning rate, and optimization. It effectively detects COVID-19 coughs and other types of coughs.

Extreme Gradient Boosting (XGBoost) classifier is a decision-tree-based ensemble ML technique that utilizes a gradient boosting structure. This advanced and powerful technique can deal with data irregularities and further reduce overfitting [47]. Some previous studies have reported the performance of the XGBoost classifier in detecting COVID-19 in cough samples

Gradient Boosting (GBoost) generates an additive model according to the forwarding stage-wise, and summarizes it by optimizing the differentiable loss function [48]. At each stage, regression trees (equal to the total number of classes) are fitted to the negative gradient of the binomial or multinomial deviation loss function.

Logistic Regression (LR) is a parametric classification model with fixed parametric numbers that predict categorical or discrete output for given input features. We can use multinomial logistic regression in scenarios with multiple categories rather than two categories Madhurananda et al. successfully used it for COVID-19 cough detection.

k-Nearest Neighbor (k-NN) is a well-known classifier that appears in large-scale ML applications. As we have seen from previous studies, researchers used k-NN in non-COVID-19 applications such as night coughing and sniffing [50] and used k-NN to detect COVID-19 in cough samples

Histogram-based Gradient Boosting (HGBoost) is a highly desirable ML technology, where the application needs to get better quality performance in less inference time. The main advantage of histogram-based gradient boosting technology is speed. Chung et al. [52] successfully explored this method to predict the severity of COVID-19.

2.4. Training strategies and hyper-parameters optimization

To assess the efficacy of various components of the suggested strategy, we provide three training strategies, namely training strategies 1, 2, and 3. The dataset makes it clear that the COVID-19 positive category is underrepresented, which could have a negative impact on the ML classifier's performance. Therefore, to balance the dataset during training and improve the performance of the ML classifier, we applied the Synthetic Minority Oversampling Technique (SMOTE). The key distinction between training strategies 1 and 2 is that although strategy 1 does not implement SMOTE during training, strategy 2 does. They both, however, employ the identical hyper-parameters. Contrarily, method 3 differs from strategies 1 and 2 in that it incorporates layered cross-validation with hyper-parameters optimization. The nested cross-validation incorporates a 5-fold stratified cross-validation inner loop for hyper-parameter optimization and a 10-fold stratified cross-validation outer loop for the SMOTE training process. There is a list of the hyper-parameters utilised in empirical optimization evaluation. The default criterion (i.e., 0.50) results in subpar performance for classes where we experience class imbalance issues. Therefore, we alter the probability threshold that specifies the probability to the class label using the threshold moving technique. In our trials, we calculate ROC-AUC scores for each fold of cross-validation by incrementing 0.001, and then choose the optimum threshold that results in the highest ROC-AUC score shows different configurations of training strategies. For training techniques 1 and 2, all classifiers are trained using fixed hyper-parameters. We employ 10 fold cross-validation in both approaches, dividing the dataset into a train set and a test set. Utilizing layered cross-validation, approach 3 separates the dataset into train and test sets using outer cross-validation. We use grid search in the inner loop of cross-validation to find the optimal parameters based on the training set. The outer loop of cross-validation is created by training classifiers with the best parameters using the same train set after obtaining them. The proposed model is then put to the test on a brand-new test set.

2.1.1 BLOCK DIAGRAM

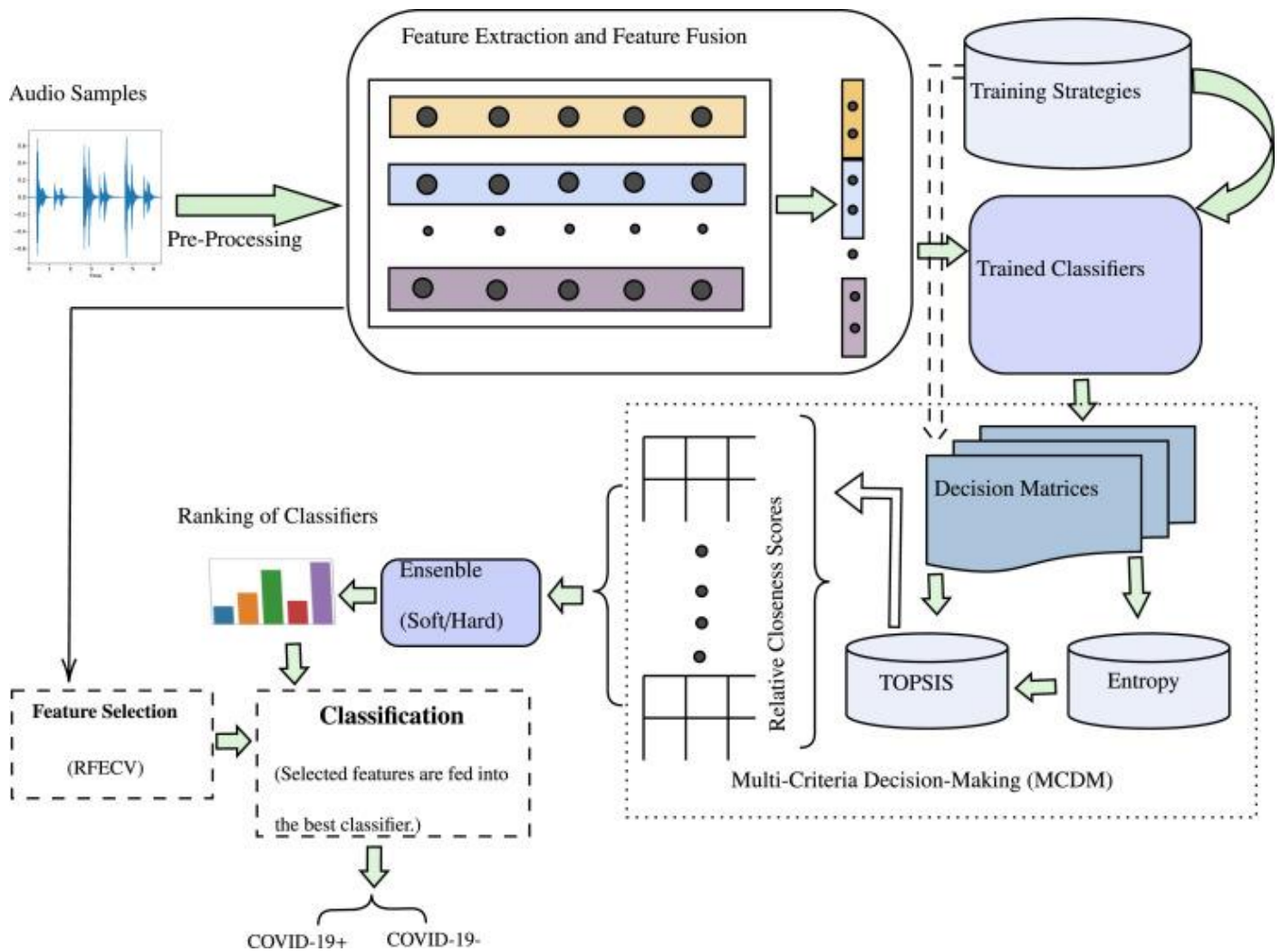


Fig 2.1.2.a: Block Diagram

2.1.4 ARCHITECTURE

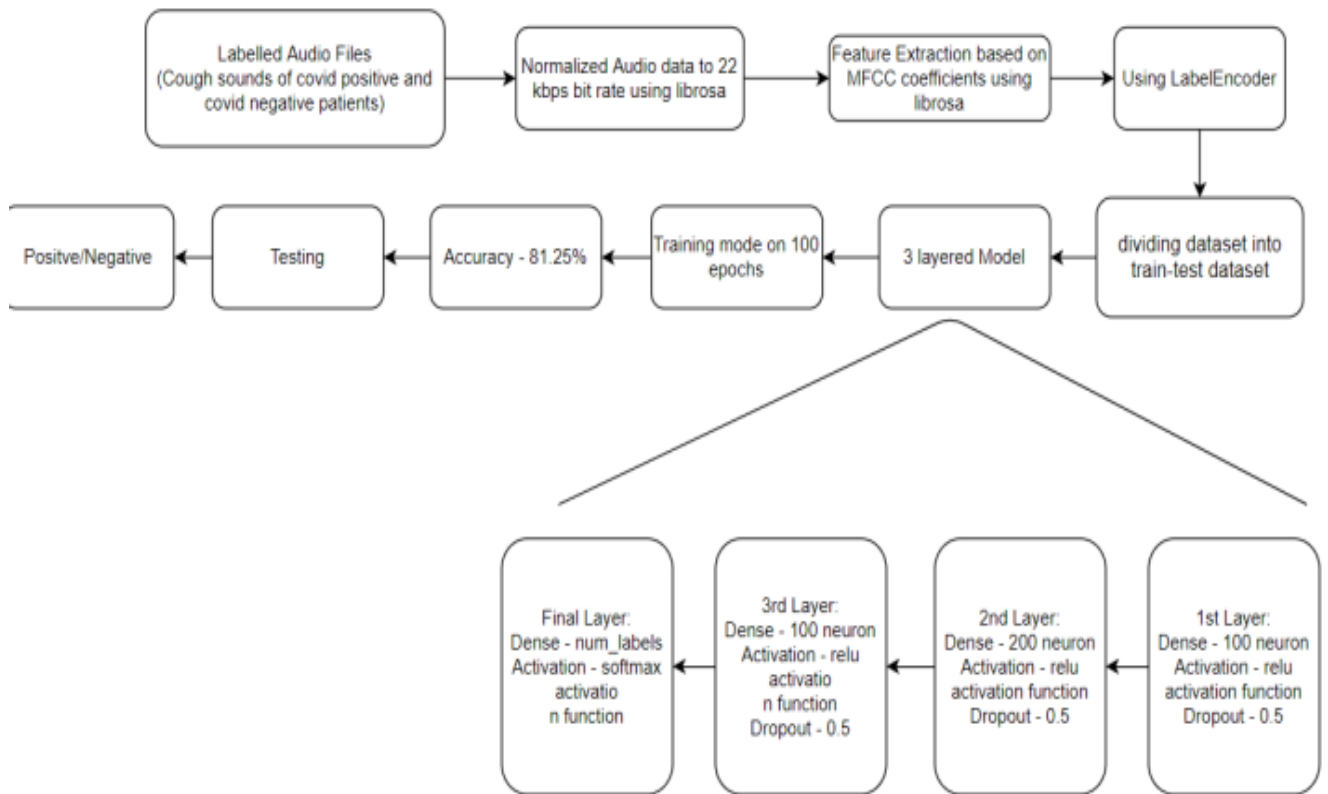


Fig 2.1.4.a: Architecture

Overview of Software

TensorFlow allows developers to create *dataflow graphs*—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or *tensor*.

TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

All of information is made available to programmers by TensorFlow using the Python language. Python offers practical ways to define how high-level abstractions might be coupled together that are simple to understand and use. TensorFlow is supported on Python versions 3.7 through 3.10; previous versions of Python may also work, but this is not guaranteed.

TensorFlow applications are Python apps, while its nodes and tensors are Python objects. But Python isn't used to carry out the math operations. The transformation libraries that TensorFlow offers are written as high-performance C++ binaries. Python only manages the communication between the parts and offers high-level programming abstractions to connect them.

The Keras library is used for high-level tasks in TensorFlow, such as generating nodes and layers and connecting them. The Keras API appears to be straightforward; a simple model with three layers can be defined in less than 10 lines of code, and the training code for the same model only requires a few additional lines. However, you can "lift the hood" and perform more minute work, such creating your own training loop.

Summary

The main reason for the spread of Coronavirus can be the inadequacy of test kits, as well as the heavy cost and loss of time in determining clinical test results. In this study, we tried to find a solution to the COVID-19 epidemic using an open-access dataset and machine learning technology. Based on this technology, we have developed a model that can classify COVID-19 cough records obtained from smartphones. We knew that the dry cough symptom is an important factor in the diagnosis of COVID-19. Thus, we created the main framework of this research by taking cough-based studies into consideration. The most important goal was to provide a virtual test opportunity away from the clinical and hospital environment based on machine learning technology. Cough records of 16 subjects suspected of COVID-19 were analysed to create the dataset. The obtained cough sounds became a ready-made dataset by the pre-processing and segmentation process. Considering the spectrogram graph based on STFT, we focused on effective features. Focusing on the PSD difference of negative and positively labelled cough sounds, strong features were prepared from dominant time and frequency ranges for the STFT feature extraction method. In addition, the MFCC feature extraction method was included in the analysis for the diagnosis of COVID-19 cough. Based on the work done for the diagnosis of COVID-19 cough sound, the MFCC feature extraction technique and RBF kernel SVM were selected as the method with the best percentage of success.

In the future, a stronger cough-based COVID-19 diagnostic study can be conducted by a larger dataset. Future studies can also increase the number of subjects and the classification accuracy using different feature extraction and classification methods.

CHAPTER – 3

RESULT AND ANALYSIS / TESTING

3.1 JUPYTER Implementation

```
#library import
import pandas as pd
import librosa
import librosa.display
import IPython.display as ipd
import numpy as np
from tqdm import tqdm
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

The above Snippet is used to import required packages in jupyter.

```
#Read the csv file

audio_dataset_path =
'C:/Users/sreev/OneDrive/Desktop/coughwavcovid/'
metadata_path =
pd.read_csv('C:/Users/sreev/OneDrive/Desktop/coughwavcovid/cough_co
vid.csv')

#Show data
metadata_path.head(30)
```

The above snippet is used to import dataset consists of cough signals.

```
#we extract 100 characteristics from each sound file
def audio_feature_extraction(file):
    audio, sample_rate =
librosa.load(audio_dataset_path+file_name, res_type='kaiser_fast')
    mfcc_feat = librosa.feature.mfcc(y=audio, sr=
sample_rate, n_mfcc=100)
    mfcc_scaled_features = np.mean(mfcc_feat.T, axis =0)
    return mfcc_scaled_features

import numpy as np
extracted_features = []
for index_num, row in tqdm(metadata_path.iterrows()):
    file_name = str(row['file_name'])
    final_class_labels = str(row['class'])
    data = audio_feature_extraction(file_name)
    extracted_features.append([data, final_class_labels])
```

This code is used to link the audio file with csv file what we have created already. And it is

used to find MFCC coefficients which were used through the analysis of dry and wet cough detection as well as highlighted as successful features for audio analysis.

```
# Split the dataset into independent and dependent dataset
X=np.array(extracted_features_df['feature'].tolist())
y=np.array(extracted_features_df['class'].tolist())
X.shape
y
```

This code snippet is used to split the available dataset into independent and dependent variables. In this project independent variable is feature and dependent variable is class (positive or negative) based on the feature.

```
#Select training and test data from the model
labelencoder=LabelEncoder()
y=to_categorical(labelencoder.fit_transform(y))
### Train Test Split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_s
tate=42)
```

In above code snippet we split the dataset into training and testing dataset. We are giving 80% of data to training because more training leads to more accuracy and 20% of data to testing.

```
#Show the data
print("Training data:")
print(X_train.shape)
print(y_train.shape)
```

```
print("Test data:")
print(X_test.shape)
print(y_test.shape)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Dense,Activation, Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics
## No of classes
num_labels=y.shape[1]
print(num_labels)
```

```

#Building the layers
model=Sequential()
###first layer
model.add(Dense(100,input_shape=(100,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###second layer
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###third layer
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))

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

## Training
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime
num_epochs = 400
num_batch_size = 10
#save the training
checkpointer = ModelCheckpoint(filepath='cought_covid.h5',
                               verbose=1, save_best_only=True)

start = datetime.now()
print("Start Training...")
history=model.fit(X_train,          y_train,          batch_size=num_batch_size,
epochs=num_epochs,          validation_data=(X_test,          y_test),
callbacks=[checkpointer], verbose=1)
duration = datetime.now() - start
print("Training completed in time: ", duration)

```

In above snippet we used ModelCheckpoint which used in conjunction with training using model. fit()

to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved. And we are training the model in above snippet using `model.fit`.

```
#Show the graph accuray
test_accuracy=model.evaluate(X_test,y_test,verbose=0)
print(test_accuracy[1])

import matplotlib.pyplot as plt
plt.figure(figsize=[14,10])
plt.subplot(211)
plt.plot(history.history['loss'],'r',linewidth=3.0)
plt.plot(history.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves',fontsize=16)

# Accuracy Curves
plt.figure(figsize=[14,10])
plt.subplot(212)
plt.plot(history.history['accuracy'],'r',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)
```

In above code snippet we calculated accuracy using `model.evaluate`. And we are plotting loss curves by comparing training loss and validation loss and also plotting accuracy by comparing training accuracy and validation accuracy.

```

# function to extract features from the audio file
# transform each category with it's respected label
def extract_feature(file_name):
    # load the audio file
    audio_data, sample_rate = librosa.load(file_name,
res_type='kaiser_fast')

    # get the feature
    feature = librosa.feature.mfcc(y=audio_data, sr=sample_rate,
n_mfcc=100)

    # scale the features
    feature_scaled = np.mean(feature.T,axis=0)

    # return the array of features
    return np.array([feature_scaled])

# function to predict the feature

def print_prediction(file_name):

    # extract feature from the function defined above
    prediction_feature = extract_feature(file_name)

    # get the id of label using argmax
    predicted_vector = np.argmax(model.predict(prediction_feature), axis=-
1)

    # get the class label from class id
    predicted_class = labelencoder.inverse_transform(predicted_vector)

    # display the result
    print("The predicted class is:", predicted_class[0], '\n')

```

Here we are defining extract_feature function for extracting MFCC coefficients for test data set and defining another function print_prediction for prediction of class (positive or negative)for test data set.

```
# File name=file to predict
#upload files that have not been used in the model
file_name ='C://Users//sreev//Downloads//sample-38.wav'
# get the output
print_prediction(file_name)
```

Here we are predicting the class for test data set using previously defined function print_prediction.

```
# play the file
dat1, sampling_rate1 = librosa.load(file_name)
plt.figure(figsize=(20, 10))
D = librosa.amplitude_to_db(np.abs(librosa.stft(dat1)), ref=np.max)
plt.subplot(4, 2, 1)
librosa.display.specshow(D, y_axis='linear')
librosa
plt.colorbar(format='%+2.0f dB')
plt.title(file_name)
ipd.Audio(file_name)
```

The above code is used to plot spectrogram and for playing audio file.

3.2 JUPYTER Output Screenshots

	feature	class
30	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
31	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
32	[-411.40427, 16.092903, -32.722694, -5.9505596...	Negative
33	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
34	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
35	[-411.40427, 16.092903, -32.722694, -5.9505596...	Negative
36	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
37	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
38	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative
39	[-411.40427, 16.092907, -32.7227, -5.9505563, ...	Negative

Fig 3.2.a: Features for each class

```
: #Show the data
print("Training data:")
print(X_train.shape)
print(y_train.shape)

print("Test data:")
print(X_test.shape)
print(y_test.shape)
```

```
Training data:
(32, 100)
(32, 2)
Test data:
(8, 100)
(8, 2)
```

Fig 3.2.b: showing dimensions of training and testing data

```

Epoch 397: val_loss did not improve from 0.00000
4/4 [=====] - 0s 15ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 4.5150e-06 - val_accuracy: 1.0000
Epoch 398/400
1/4 [=====>.....] - ETA: 0s - loss: 0.0013 - accuracy: 1.0000
Epoch 398: val_loss did not improve from 0.00000
4/4 [=====] - 0s 13ms/step - loss: 6.5535e-04 - accuracy: 1.0000 - val_loss: 4.2915e-06 - val_accuracy: 1.0000
Epoch 399/400
1/4 [=====>.....] - ETA: 0s - loss: 0.0563 - accuracy: 1.0000
Epoch 399: val_loss did not improve from 0.00000
4/4 [=====] - 0s 17ms/step - loss: 0.0692 - accuracy: 0.9688 - val_loss: 6.4819e-06 - val_accuracy: 1.0000
Epoch 400/400
1/4 [=====>.....] - ETA: 0s - loss: 0.0057 - accuracy: 1.0000
Epoch 400: val_loss did not improve from 0.00000
4/4 [=====] - 0s 14ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 1.0282e-05 - val_accuracy: 1.0000
Training completed in time: 0:00:42.593785

```

Fig 3.2.c: Training of data

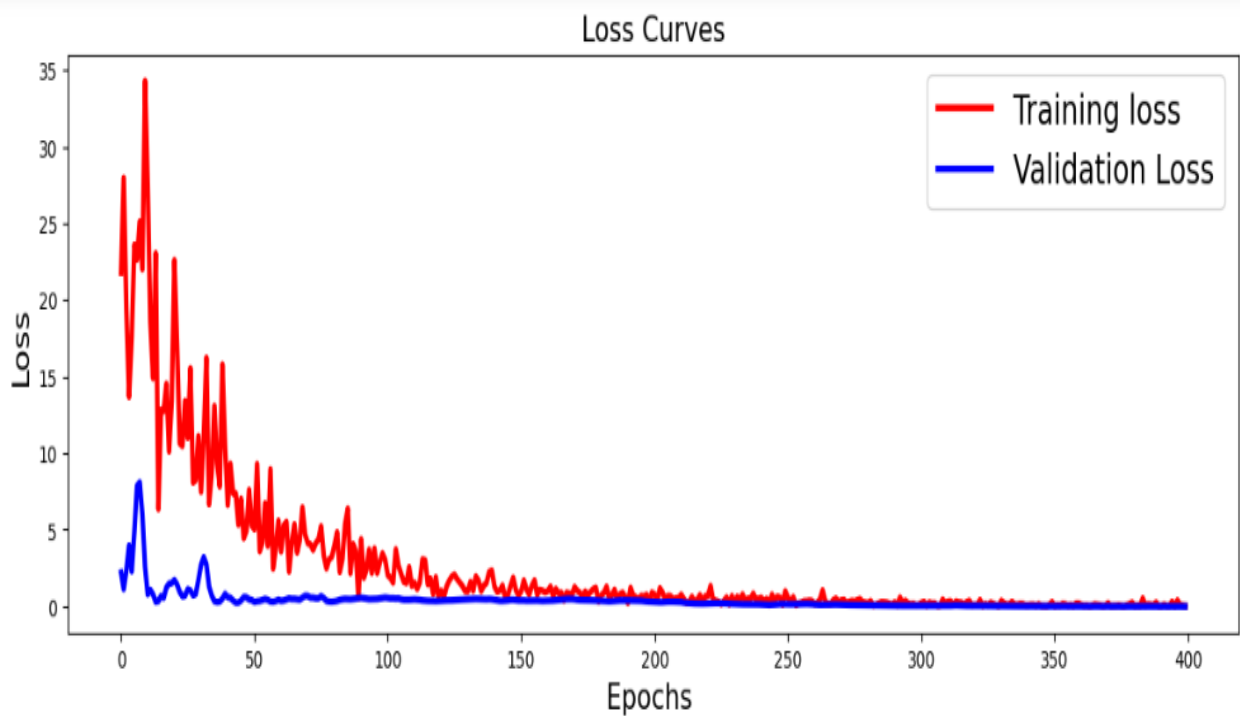


Fig 3.2.d: Loss Curves

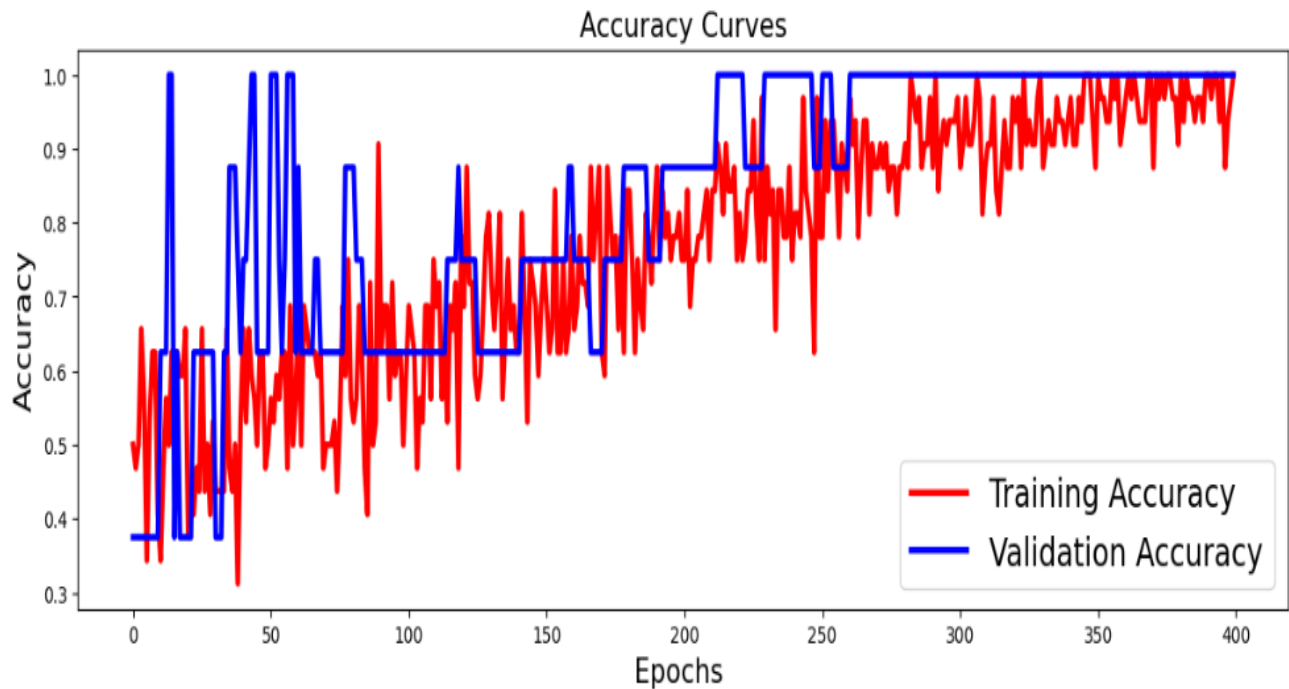


Fig 3.2.e: Accuracy curves

```
In [92]: # File name=file to predict
#upload files that have not been used in the model

file_name = 'C:/Users/sreev/OneDrive/Desktop/coughwavcovid//sample-10N.wav'

# get the output
print_prediction(file_name)

# play the file

dat1, sampling_rate1 = librosa.load(file_name)
plt.figure(figsize=(20, 10))
D = librosa.amplitude_to_db(np.abs(librosa.stft(dat1)), ref=np.max)
plt.subplot(4, 2, 1)
librosa.display.specshow(D, y_axis='linear')
librosa
plt.colorbar(format='%+2.0f dB')
plt.title(file_name)

ipd.Audio(file_name)

1/1 [=====] - 0s 59ms/step
The predicted class is: Negative
```

Out[92]:

▶ 0:04 / 1:56 ◀ ⋮

Fig 3.2.f: Result for Negative audio signal

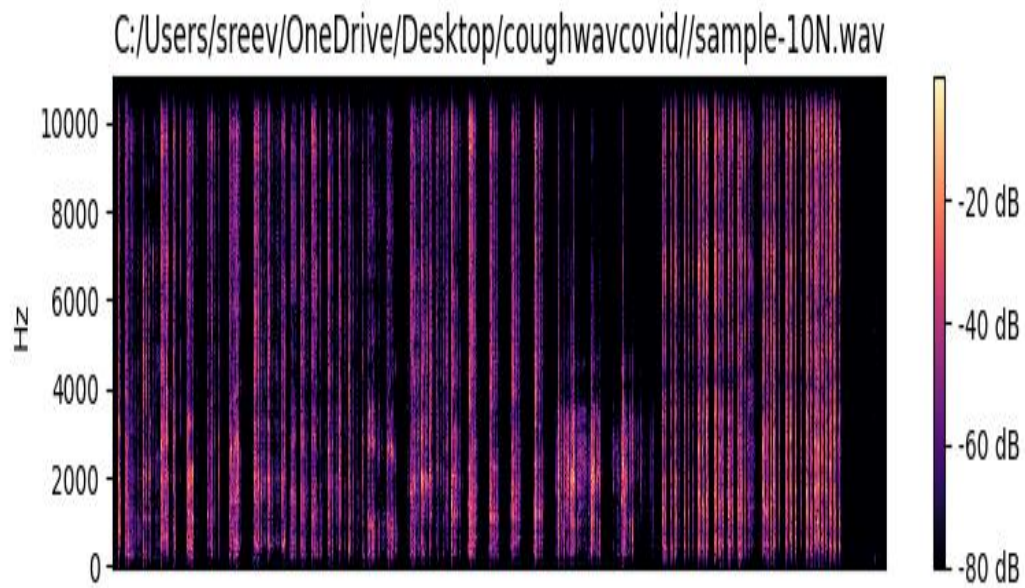


Fig 3.2.g: Spectrogram for Negative audio signal

```
In [95]: # File name=file to predict
#upload files that have not been used in the model

file_name = 'C://Users//sreev//Downloads//sample-38.wav'

# get the output
print_prediction(file_name)

# play the file

dat1, sampling_rate1 = librosa.load(file_name)
plt.figure(figsize=(20, 10))
D = librosa.amplitude_to_db(np.abs(librosa.stft(dat1)), ref=np.max)
plt.subplot(4, 2, 1)
librosa.display.specshow(D, y_axis='linear')
librosa
plt.colorbar(format='%+2.0f dB')
plt.title(file_name)

ipd.Audio(file_name)

1/1 [=====] - 2s 2s/step
The predicted class is: Positive
```

Out[95]:



Fig 3.2.h: Result for Positive audio signal

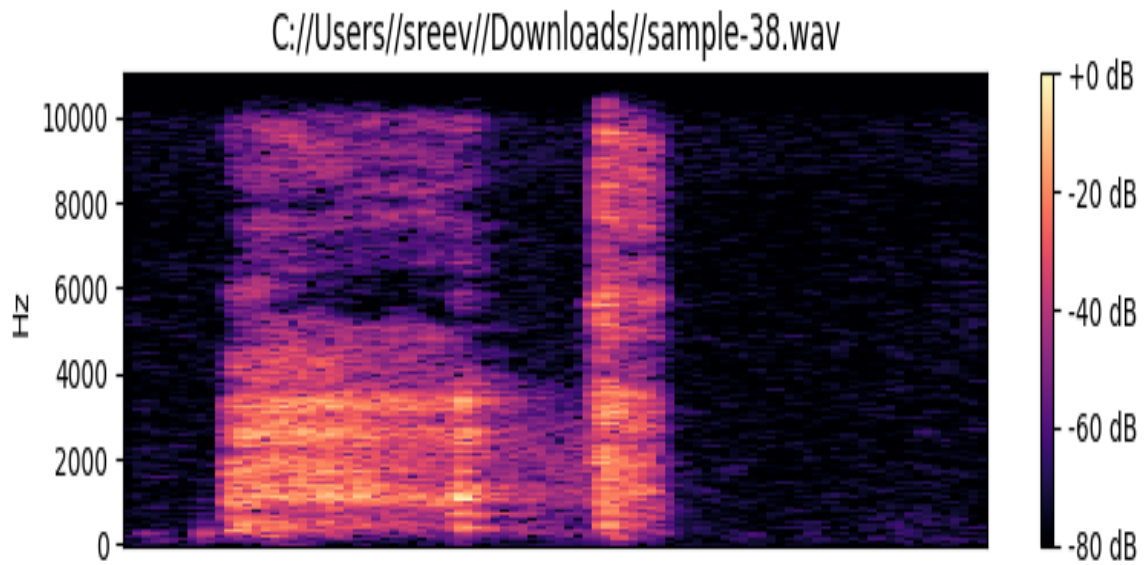


Fig 3.2.i: Spectrogram for Positive audio signal

3.3 Summary

The dataset containing the cough signals were imported into the jupyter environment and the basic pre-processing techniques were carried on to split data into independent and dependent variables. And then MFCC's coefficients were extracted which were used through the analysis of dry and wet cough detection as well as highlighted as successful features for audio analysis. After that splitting of dataset were done by giving 80 % of data to training and 20% of data to testing. Then we are building layers which are known as deep learning neural networks a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. After this training of train data set is done. And for more visualization we plotted loss curves and accuracy curves. At last test data set is tested and class is predicted and plotted respective spectrograms.

CHAPTER – 4

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

There is no longer anyone on the earth who has not been impacted by COVID-19, which has taken control. The hunt for a cure for the illness is currently more intense than ever among researchers, clinical industry researchers, and genetic scientists. One tool that could keep this endeavour to evaluate academics and scientists continuing is artificial intelligence (AI). We compare the conclusions and methods put forward by each author in the literature review section, which focuses on literature reviews for COVID-19 disorder analysis using data on COVID-19 breathing sounds and other AI-based methods.

The use of speech signals for COVID-19 identification can be a valuable and cost-effective tool because it avoids the need for time-consuming medical tests. This method, which acts as an automatic detection tool, can quickly determine a patient's initial condition without having to take them to a hospital or ask any medical personnel for assistance. In this research, we are forecasting positive or negative outcomes using the MFCC'S coefficients. This paper uses a straightforward SVM-based classifier for detection.

FUTURE SCOPE:

In future we can use more training data so that accuracy will increase. And we can propose to develop a web-based application for COVID-19 testing through the cough sounds using machine learning. It can be made with the objective of providing the testing at home hence reducing the need and the risk of going for the in-person testing, which provides a one-of-a-kind functional instrument for fast, cost-effective, and most crucially, safe monitoring, tracing, and tracking, and thereby curbing the worldwide pandemic's wild spread by essentially allowing everyone to test.

CHAPTER – 5

REFERENCES

- [1] Imran, A., Posokhova, I., Qureshi, H. N., Masood, U., Riaz, M. S., Ali, K., ... & Nabeel, M. (2020). AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app. *Informatics in Medicine Unlocked*, 20, 100378.
- [2] Ritwik, K. V. S., Kalluri, S. B., & Vijayaseenan, D. (2020). COVID-19 Patient Detection from Telephone Quality Speech Data. *arXiv preprint arXiv:2011.04299*.
- [3] Hassan, A., Shahin, I., & Alsabek, M. B. (2020, November). Covid-19 detection system using recurrent neural networks. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)* (pp. 1-5). IEEE.
- [4] Brown, C., Chauhan, J., Grammenos, A., Han, J., Hasthanasombat, A., Spathis, D., ... & Mascolo, C. (2020, August). Exploring automatic diagnosis of covid-19 from crowdsourced respiratory sound data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3474-3484).
- [5] Juliana Knocikova, J Korpas, M Vrabec, and Michal Javorka. 2008. Wavelet analysis of voluntary cough sound in patients with respiratory diseases. *Journal of physiology and pharmacology* 59 Suppl 6 (10 2008), 331–340.
- [6] Morteza Heidari, Seyedehnafiseh Mirniaharikandehei, Wei Liu, Alan B. Hollingsworth, Hong Liu, and Bin Zheng. 2020. Development and Assessment of a New Global Mammographic Image Feature Analysis Scheme to Predict Likelihood of Malignant Cases. *IEEE Transactions on Medical Imaging* 39 (4 2020), 1235–1244.
- [7] Joseph R. Larsen, Margaret R. Martin, John D. Martin, Peter Kuhn, and James B. Hicks. 2020. Modeling the Onset of Symptoms of COVID-19. *Frontiers in Public Health* 8 (8 2020).
- [8] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. *arXiv:cs.CL/1612.08083*.
- [9] André F. T. Martins and Ramón Fernandez Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. *arXiv:cs.CL/1602.02068*
- [10] Belkacem A, Ouhbi S, Lakas A (2021) End-to-end AI-based point-of-care diagnosis system for classifying respiratory illnesses and early detection of COVID-19: a theoretical framework. *Front Med* 8(2):1–13
- [11] Alqudaihi K, Aslam N, Khan I (2021) Cough sound detection and diagnosis using artificial intelligence techniques: challenges and opportunities. *IEEE Access* 9:102327–102344
- [12] <https://www.kaggle.com/code/josluisandreu/covid-cough-classification/data>

CHAPTER – 6

BIO DATA



NAME: P SREEVARDHAN
PHONE: 9392757212
EMAIL: pemmareddy.sreevardhan2020@vitstudent.ac.in
ADRESS: Rajagopala puram, NAIDUPETA,
tirupati, Andhra pradesh



NAME: M.V.SAI JASWANTH
PHONE: 9573324825
EMAIL: Venkata.saijaswanth2020@vitstudent.ac.in
ADRESS: 204-A
Pavani prestage apartment maguntalayout
Nellore Andhra Pradesh



NAME: N G JAGADEESWAR
PHONE: 8431010790
EMAIL: Govind.jagadeeswar2020@vitstudent.ac.in
ADRESS: ramurthy nagar
Bengalore , Karnataka , 600056

