In [1]:

```
%load_ext nb_black
```

Import Section

In [3]:

```
import os
import json
import numpy as np
import matplotlib.pyplot as plt
import re
import pandas as pd
import librosa
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Input
from sklearn.model_selection import train_test_split
import IPython.display as ipd
from tensorflow.keras.models import load_model
from sklearn.preprocessing import LabelEncoder
```

In [4]:

```
DATASET_PATH = "COVID-19"
```

In [5]:

```
files = []

# r=root, d=directories, f = files

for r, d, f in os.walk(DATASET_PATH):
    for file in f:
        if ".wav" in file:
            files.append(os.path.join(r, file))
```

In [6]:

```
files[:9]
```

Out[6]:

```
['COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-o.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-a.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breathing-shallow.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cough-shallow.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-e.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cough-heavy.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breathing-deep.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/counting-normal.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/counting-fast.wav']
```

Feature Extraction

In [8]:

```
RMSE = []
chroma_stft = []
spec_cent = []
spec_bw = []
rolloff = []
zcr = []
user_id = []
mfcc = []

for f in files:

    y, sr = librosa.load(f, sr=None)

    if y is None or len(y) == 0:
        continue

    else:
        user_id.append(f.split("/")[1])

        metadata = json.load(open("/".join(f.split("/")[:2]) + "/metadata

        # Root Mean Squared Error
        RMSE.append(np.mean(librosa.feature.rms(y)[0]))

        # Chroma Based Short Time Fourier Transform
        chroma_stft.append(np.mean(librosa.feature.chroma_stft(y=y, sr=sr

        # Spectral Centroid
        spec_cent.append(np.mean(librosa.feature.spectral_centroid(y=y, s

        # Spectral Bandwidth
        spec_bw.append(np.mean(librosa.feature.spectral_bandwidth(y=y, sr

        # Spectral RollOff
        rolloff.append(np.mean(librosa.feature.spectral_rolloff(y=y, sr=s

        # Zero Crossing Rate
        zcr.append(np.mean(librosa.feature.zero_crossing_rate(y)))

        mfccs = librosa.feature.mfcc(y=y, sr=sr)
        m = []

        for e in mfccs:
            m.append(np.mean(e))
        mfcc.append(m)
```

```
/home/manthan/Drive/Work/VSCode/vscode/lib/python3.8/site-packages/lib
rosa/core/pitch.py:153: UserWarning: Trying to estimate tuning from em
pty frequency set.
  warnings.warn("Trying to estimate tuning from empty frequency set.")
```

In [9]:

```
data1 = pd.DataFrame(
    {
        "user_id": user_id,
        "filepath": files,
        "RMSE": RMSE,
        "chroma_stft": chroma_stft,
        "spec_cent": spec_cent,
        "spec_bw": spec_bw,
        "spec_rolloff": rolloff,
        "zcr": zcr,
    }
)
```

In [10]:

```
cols = ["mfcc_" + str(i) for i in range(1, 21)]
data2 = pd.DataFrame(mfcc, columns=cols)
data2["user_id"] = user_id
```

In [11]:

```
data1.head()
```

Out[11]:

	user_id	filepath	RMSE	ch
0	7DfMFXPDu3W2Fxjs8w0OsLIY8em1	COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-o.wav	0.118379	
1	7DfMFXPDu3W2Fxjs8w0OsLIY8em1	COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-a.wav	0.144788	
2	7DfMFXPDu3W2Fxjs8w0OsLIY8em1	COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breathin...	0.000135	
3	7DfMFXPDu3W2Fxjs8w0OsLIY8em1	COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cough-sh...	0.018265	
4	7DfMFXPDu3W2Fxjs8w0OsLIY8em1	COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-e.wav	0.107049	

In [12]:

```
data2.head()
```

Out[12]:

	mfcc_1	mfcc_2	mfcc_3	mfcc_4	mfcc_5	mfcc_6	mfcc_7	mfcc_8
0	-363.377258	144.358337	27.980934	15.131627	-4.168990	-31.025049	-20.511789	-20.623474
1	-345.390533	124.867485	-4.307336	19.490623	7.525982	-17.402979	6.819692	-16.135691
2	-860.161987	97.513298	-21.191671	8.036111	6.223122	-6.836332	8.026308	2.952522
3	-525.321655	51.241253	-15.734446	-2.986843	5.281782	-3.511705	3.455106	-0.633038
4	-406.449951	121.712585	16.063465	29.269457	18.560997	-3.910539	12.661272	-5.819675

5 rows × 21 columns

In [32]:

```
data = pd.concat([data1, data2], axis=1)
```

In [33]:

```
data.shape
```

Out[33]:

```
(3798, 29)
```

In [34]:

```
data.head()
```

Out[34]:

filepath	RMSE	chroma_stft	spec_cent	spec_bw	spec_rolloff	
COVID- N2Fxjs8w0OsLIY8em1/vowel- o.wav	0.118379	0.253389	1390.837727	1459.099553	2264.121835	0.034
COVID- N2Fxjs8w0OsLIY8em1/vowel- a.wav	0.144788	0.246714	1774.427666	2057.234269	3452.113464	0.020
COVID- =xjs8w0OsLIY8em1/breathin...	0.000135	0.495409	5502.609625	5720.697350	11688.208532	0.051
COVID- V2Fxjs8w0OsLIY8em1/cough- sh...	0.018265	0.434287	4009.550598	4088.996725	7927.490831	0.065
COVID- N2Fxjs8w0OsLIY8em1/vowel- e.wav	0.107049	0.297738	1623.703648	2021.015855	3080.055950	0.024

In [35]:

```
file_le = LabelEncoder()
file_le.classes_ = np.load("file_le.npy", allow_pickle=True)

data["filename"] = data["filepath"].apply(lambda x: x.split("/")[2])
data["filename"] = file_le.transform(data["filename"])

data.drop(["user_id"], inplace=True, axis=1)

# target data
cs_le = LabelEncoder()
cs_le.classes_ = np.load("cs_le.npy", allow_pickle=True)

target_data = np.load("target_data.npy")
```

In [36]:

```
data.drop(["filepath"], inplace=True, axis=1)
```

In [37]:

```
# data.to_csv("mfcc_and_features.csv", index=False)
```

In [38]:

```
data.head()
```

Out[38]:

...	mfcc_12	mfcc_13	mfcc_14	mfcc_15	mfcc_16	mfcc_17	mfcc_18	mfcc_19
...	-14.310049	1.837768	-9.357093	-16.901194	-8.871928	-11.130017	-10.805934	-3.902681
...	-20.900917	-10.857018	-14.837809	-15.201618	-5.872452	-1.326357	-1.137605	-7.114715
...	-7.578856	-0.824565	-0.376975	-6.388448	-2.045947	0.254025	-0.853397	-0.025326
...	-4.711785	-0.084784	-2.399943	-2.744585	0.439062	-1.520346	-1.996574	0.463630
...	-16.488800	-6.080915	-11.323594	-14.311434	-7.020574	-10.257932	-9.524367	-6.419607

In [39]:

```
fully_featured_data = data.values.reshape((422, 9, 27))
```

In [40]:

```
fully_featured_data[0][0]
```

Out[40]:

```
array([ 1.18379205e-01,  2.53389359e-01,  1.39083773e+03,  1.45909955e
+03,
        2.26412184e+03,  3.46209949e-02, -3.63377258e+02,  1.44358337e
+02,
        2.79809341e+01,  1.51316271e+01, -4.16899014e+00, -3.10250492e
+01,
       -2.05117893e+01, -2.06234741e+01, -2.32857609e+01, -1.53525877e
+01,
       -2.16012230e+01, -1.43100491e+01,  1.83776784e+00, -9.35709286e
+00,
       -1.69011936e+01, -8.87192822e+00, -1.11300173e+01, -1.08059340e
+01,
       -3.90268111e+00, -4.77074718e+00,  8.00000000e+00])
```

In [41]:

```
data.iloc[0]
```

Out[41]:

```
RMSE               0.118379
chroma_stft        0.253389
spec_cent       1390.837727
spec_bw         1459.099553
spec_rolloff    2264.121835
zcr                0.034621
mfcc_1          -363.377258
mfcc_2           144.358337
mfcc_3            27.980934
mfcc_4            15.131627
mfcc_5            -4.168990
mfcc_6           -31.025049
mfcc_7           -20.511789
mfcc_8           -20.623474
mfcc_9           -23.285761
mfcc_10          -15.352588
mfcc_11          -21.601223
mfcc_12          -14.310049
mfcc_13            1.837768
mfcc_14           -9.357093
mfcc_15          -16.901194
mfcc_16           -8.871928
mfcc_17          -11.130017
mfcc_18          -10.805934
mfcc_19           -3.902681
mfcc_20           -4.770747
filename           8.000000
Name: 0, dtype: float64
```

In [43]:

```
np.save("fully_featured_data.npy", fully_featured_data)
```

Data Preparation

- Order of Features Must Be Same
  - RMSE
  - Chroma_Stft
  - Spec_cent
  - Spec_bw
  - Spec_rolloff
  - Zcr
  - MFCCs (1 to 20)
  - Filename (filename => file_le.fit_transform)
- Target:
  - covid_status (covid_status => cs_le.fit_transform)

File Information

- **fully_featured_data.npy** : our final feature array of shape (422 ,9 ,27)

- **targegt_data.npy** : final target array of shape (422 ,1)
- **cs_le.npy** : covid_status label encoder classes
- **file_le.npy** : filename label encoder classes

Deep Learning Model

In [44]:

```
n_classes = len(cs_le.classes_)
```

In [69]:

```
model = Sequential()

model.add(Input(shape=fully_featured_data[0].shape))
model.add(Flatten())

model.add(Dense(units=16, activation="relu"))
model.add(Dense(units=32, activation="relu"))
model.add(Dense(units=64, activation="relu"))
model.add(Dense(units=128, activation="relu"))
model.add(Dense(units=512, activation="relu"))
model.add(Dense(units=n_classes, activation="softmax"))
```

In [70]:

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_4 (Flatten)          (None, 243)               0         
_________________________________________________________________
dense_24 (Dense)             (None, 16)                3904      
_________________________________________________________________
dense_25 (Dense)             (None, 32)                544       
_________________________________________________________________
dense_26 (Dense)             (None, 64)                2112      
_________________________________________________________________
dense_27 (Dense)             (None, 128)               8320      
_________________________________________________________________
dense_28 (Dense)             (None, 512)               66048     
_________________________________________________________________
dense_29 (Dense)             (None, 7)                 3591      
=================================================================
Total params: 84,519
Trainable params: 84,519
Non-trainable params: 0
_________________________________________________________________
```

In [71]:

```
model.compile(
    optimizer="adam",
    loss=tensorflow.keras.losses.SparseCategoricalCrossentropy(),
    metrics=["accuracy"],
)
```

Train Test Split

In [72]:

```
X_train, X_test, y_train, y_test = train_test_split(
    fully_featured_data, target_data, test_size=0.20, random_state=0
)
```

In [73]:

```
X_train.shape, X_test.shape
```

Out[73]:

```
((337, 9, 27), (85, 9, 27))
```

In [74]:

```
y_train = np.array(y_train)
y_test = np.array(y_test)
y_train.shape, y_test.shape
```

Out[74]:

```
((337,), (85,))
```

Model Training

In [75]:

```
model.fit(X_train, y_train, batch_size=1, epochs=20)
```

```
Epoch 1/20
337/337 [==============================] - 0s 540us/step - loss: 16.
6887 - accuracy: 0.5460
Epoch 2/20
337/337 [==============================] - 0s 535us/step - loss: 1.4
820 - accuracy: 0.6409
Epoch 3/20
337/337 [==============================] - 0s 506us/step - loss: 1.1
855 - accuracy: 0.6647
Epoch 4/20
337/337 [==============================] - 0s 489us/step - loss: 1.1
755 - accuracy: 0.6677
Epoch 5/20
337/337 [==============================] - 0s 490us/step - loss: 1.1
653 - accuracy: 0.6706
Epoch 6/20
337/337 [==============================] - 0s 496us/step - loss: 1.1
726 - accuracy: 0.6706
Epoch 7/20
337/337 [==============================] - 0s 485us/step - loss: 1.1
696 - accuracy: 0.6677
Epoch 8/20
337/337 [==============================] - 0s 499us/step - loss: 1.1
634 - accuracy: 0.6706
Epoch 9/20
337/337 [==============================] - 0s 483us/step - loss: 1.1
761 - accuracy: 0.6677
Epoch 10/20
337/337 [==============================] - 0s 514us/step - loss: 1.1
795 - accuracy: 0.6677
Epoch 11/20
337/337 [==============================] - 0s 571us/step - loss: 1.1
781 - accuracy: 0.6677
Epoch 12/20
337/337 [==============================] - 1s 3ms/step - loss: 1.172
7 - accuracy: 0.6677
Epoch 13/20
337/337 [==============================] - 1s 3ms/step - loss: 1.169
3 - accuracy: 0.6677
Epoch 14/20
337/337 [==============================] - 1s 3ms/step - loss: 1.179
6 - accuracy: 0.6677
Epoch 15/20
337/337 [==============================] - 1s 3ms/step - loss: 1.179
3 - accuracy: 0.6677
Epoch 16/20
337/337 [==============================] - 1s 3ms/step - loss: 1.173
9 - accuracy: 0.6677
Epoch 17/20
337/337 [==============================] - 1s 2ms/step - loss: 1.173
1 - accuracy: 0.6677
Epoch 18/20
337/337 [==============================] - 0s 514us/step - loss: 1.1
781 - accuracy: 0.6677
Epoch 19/20
337/337 [==============================] - 0s 542us/step - loss: 1.1
```

```
797 - accuracy: 0.6677
Epoch 20/20
337/337 [==============================] - 0s 564us/step - loss: 1.1
755 - accuracy: 0.6677
```

Out[75]:

```
<tensorflow.python.keras.callbacks.History at 0x7fb57c18f8e0>
```

In [76]:

```
model.evaluate(X_test, y_test, batch_size=1)
```

```
85/85 [==============================] - 0s 353us/step - loss: 1.1665
- accuracy: 0.6941
```

Out[76]:

```
[1.1665338277816772, 0.6941176652908325]
```

In [77]:

```
model.save("model_3_68_mfcc_with_other_features.h5")
```

Model Acccuracy Comparision:

- `Simple Feature Extraction` : 0.7058823704719543
- `MFCCs Feature Extraction` : 0.7176470756530762
- `MFCCs + Simple Features` : 0.6941176652908325