

# Speaker identification

Sher singh (4780568)

Hisham Unniyankal (5049651)

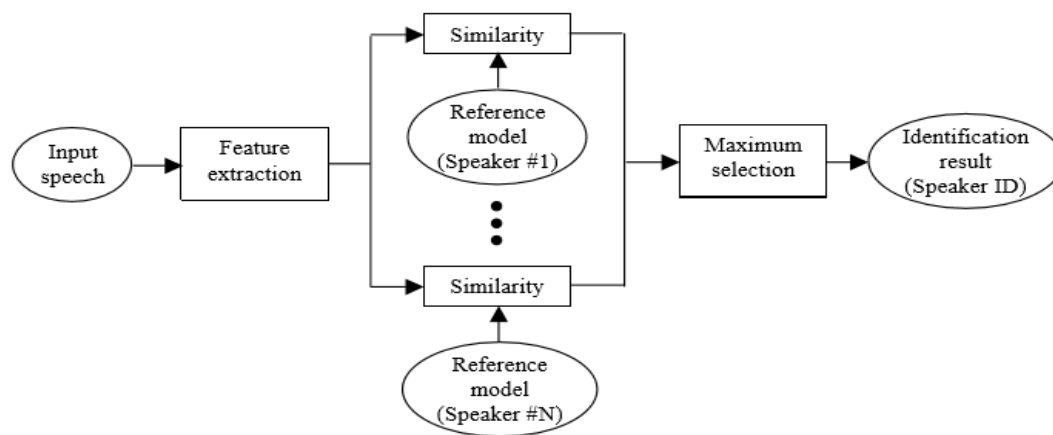
## **ABSTRACT**

The project aims to identify/recognize a speaker based on the characteristics of their speech signal. We have performed this project in two techniques. In this project, we are mostly focused on implementing MFCC (Mel Frequency Cepstral Coefficient (MFCC)) and SVM (Support Vector Machines) in pair to complete our target. we used GUI in this method and Classify speakers using Fast Fourier Transform (FFT) and 1D CNN (Convolutional neural network) with GUI. We measured MFCC with “tuned parameters” as the main feature and delta- MFCC as minor feature. And, we have implemented SVM with some tuned parameters to train our model. On the other side categorize speakers from the frequency domain representation of speech recordings, obtained via Fast Fourier Transform (FFT). In the first part, we will use librosa library for extracting MFCC features and sklearn to classify the speakers. In second part, we will use TensorFlow keras library. We include background noise to these samples to augment our data We have performed this project on <https://www.kaggle.com/kongaevans/speaker-recognition-dataset>. The GUI make the prediction easier for us. The GUI is made using tkinter library of python and we use trained models for both SVM and CNN. We used tensorflow backend to store the CNN model and pickle to save SVM model.

**Keywords:** Speaker Identification, MFCC, SVM, FFT, CNN

## INTRODUCTION

Speaker recognition is the task of detecting a speaker using their voice. speaker identification is the method of deciding which voice in a group of known voices best matches the speaker. Speaker *verification* systems are computationally less complex than speaker identification systems since they require a comparison between only one or two models, whereas speaker *identification* requires comparison of one model to  $N$  speaker models.



(a) Speaker identification

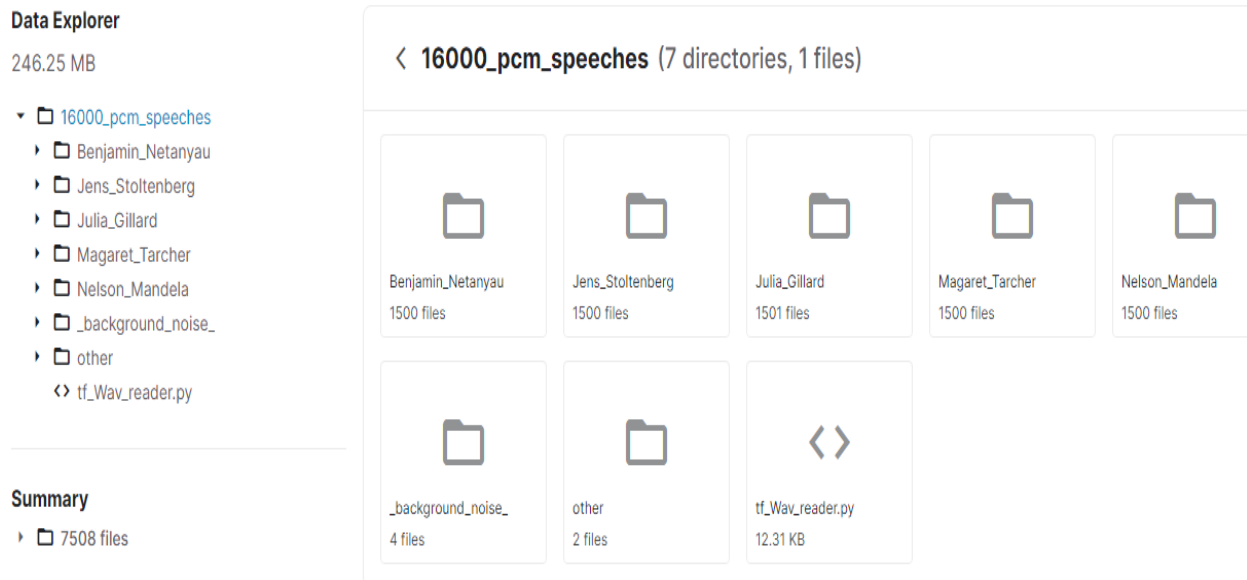
## Dataset

We download dataset from Kaggle

<https://www.kaggle.com/kongaevans/speaker-recognition-dataset>.

This dataset contains speeches of these prominent leaders; Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Tacher and Nelson Mandela which also represents the folder names. Each audio in the folder is a one-second 16000 sample rate PCM encoded.

A folder called background noise covers audios that are not speeches but can be found inside around the speaker environment e.g., audience laughing or clapping. It can be mixed with the speech while training



## Data organization









The dataset is arranged of seven folders, divided into two parts:

1. Speech samples, contains five files used for five different speakers. Every files covers 1500 audio files, all 1 second long and tested at 16000 Hz.
2. Background noise samples, contains two files and a complete of six files. These records are longer than one second and initially not tested at 16000 Hz, but we will resample them to 16000 Hz. Then We will use those six files to make 354 1-second-long noise samples to be used for training.









Let us sort these 2 categories into 2 folders:



- An audio folder which will cover all the per-speaker speech sample folders
- A noise folder which will cover all the noise samples

Before separating the audio and noise categories into 2 folders,

My Drive > 16000_pcm_speeches ▾ 👤					⌵	i
Name ↑	Owner	Last modified		File size		
 _background_noise_	me	Jun 11, 2021	me	—		
 Benjamin_Netanyau	me	Jun 11, 2021	me	—		
 Jens_Stoltenberg	me	Jun 11, 2021	me	—		
 Julia_Gillard	me	Jun 11, 2021	me	—		
 Magaret_Tarcher	me	Jun 11, 2021	me	—		
 Nelson_Mandela	me	Jun 11, 2021	me	—		
 other	me	Jun 11, 2021	me	—		
 tf_Wav_reader.py	me	Jan 9, 2020	me	12 KB		

After arranging, we end up with the following structure:

My Drive > downloads > 16000_pcm_speeches					My Drive > downloads > 16000_pcm_speeches > audio ▾	
Name ↑	Owner	Name ↑		Owner		
 audio	me	 Benjamin_Netanyau		me		
 noise	me	 Jens_Stoltenberg		me		
 tf_Wav_reader.py	me	 Julia_Gillard		me		
		 Magaret_Tarcher		me		
		 Nelson_Mandela		me		

My Drive > downloads > 16000_pcm_speeches > noise ▾						
Name ↑	Owner	Last modified				
 _background_noise_	me	Jun 10, 2021	me			
 other	me	Jun 10, 2021	me			

# Noise adding

In this section:

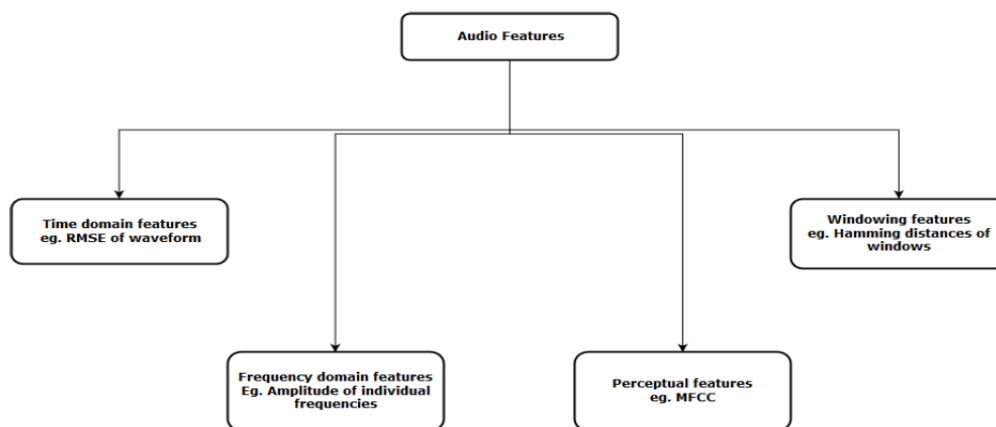
- We fill all noise samples (which must have been resampled to 16000)
- We split those noise samples to portions of 16000 samples which like 1 second duration each.

## We have done this project in two formats

1. Feature extraction with mfcc and model training with svm
2. using Fast Fourier Transform (FFT) and a 1D Convnet (cnn).

## Feature Extraction Techniques

The next step is to extract features from these audio representations, so that our algorithm can work on these features and perform the task it is designed for. In the speaker recognition problem, there are couple of features extracting techniques. Here is the visual representation of the categories of features that can be extracted from the audios.



## TYPE 1

Here we are concentrating on two main features **MFCCs** and their Derivatives, say **Delta-MFCC**. We used second order delta for features.

```
#Create dataset which contains mfcc and its delta..
def CreateFeature(signal,sr):
    """Reads and decodes an audio file to wave file."""
    #signal_trim = librosa.effects.remix(signal, intervals=librosa.effects.split(signal))
    mfcc = librosa.feature.mfcc(y=signal, sr=sr)
    mfcc_delta = librosa.feature.delta(mfcc)
    mfcc_delta2 = librosa.feature.delta(mfcc, order=2)
    combined = np.empty((20,132))
    combined[:, :44] = mfcc
    combined[:, 44:88] = mfcc_delta
    combined[:, 88:] = mfcc_delta2
    return combined
```

MFCC is one of the main feature extraction techniques for speaker identification technique. The goal of feature extraction is to find a set of properties of a sound that have acoustic correlations to the speech signal which are parameters that can somehow be computed or estimated through processing of the signal waveform. Such parameters are termed as features.

**then we set data set labels**

```
dataset,labels = CreateDataset(DATASET_AUDIO_PATH,DATASET_NOISE_PATH)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (w
if __name__ == '__main__':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (
# Remove the CWD from sys.path while we load stuff.
0 augmented data completed
1000 augmented data completed
2000 augmented data completed
3000 augmented data completed
4000 augmented data completed
5000 augmented data completed
6000 augmented data completed
7000 augmented data completed
```

## **DATA preprocessing**

We split our data set into train test. we use training dataset to train the model and test dataset to evaluate the performance of the trained model with test size=0.33, random state=42

## **Training The model**

Here, using SVM classifier from sklearn. linear library for train the model with training data.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

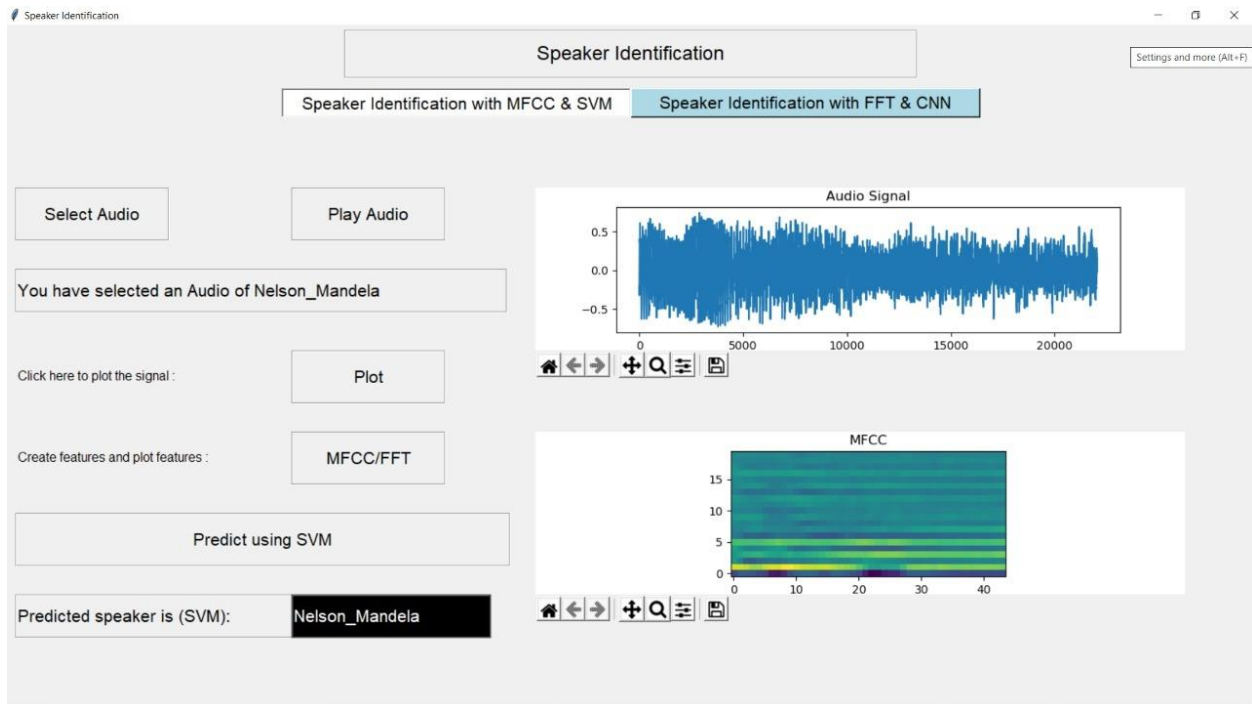
## **Model prediction**

we obtained a 98% accuracy on training set and 97% on test set

```
[ ] print("Accuracy on train set : {}".format(np.mean(ytr_pred==y_train)*100))  
    print("Accuracy on test set : {}".format(np.mean(yte_pred==y_test)*100))
```

```
Accuracy on train set : 98.58393579624594  
Accuracy on test set : 97.31349313223808
```

# We can predict speaker by uploading audio to gui by using trained model



## TYPE 2

We used CNN-1d in tensorflow keras to classify speakers from the frequency domain representation of speech recordings, obtained via Fast Fourier Transform (FFT).

It indicates the following:

- How to use tf. data to load, preprocess and feed audio streams into a model
- How to create a 1D convolutional network with remaining connections for audio classification.

Our procedure:



- We arrange a dataset of speech samples from different speakers, as well as the speaker as label.
- And then add up some background noise to augment our data.
- We take Fast Fourier transform of these samples.
- We will train our model with convnet(1d) to identify the correct speaker provided a noisy FFT speech sample.

## FFT

For speech signals analysis the common transform used called the Fast Fourier transform (FFT). FFT gives the standard representation of a speech signal in the frequency domain. Short Fourier transform be able to hold time frequency changes. The disadvantage of FFT that it is not suitable for the signals whose frequencies are time varying hence in case of FFT is believes that the signals are stationary in nature. It allocates working in frequency domain and therefore using the frequency spectrum of the speech signal as a substitute of waveform. Frequency domain gives extra information regarding the speech signal and thus can be extra useful to distinguish between speakers. Speech signal/audio processing methods initiate by converting the raw speech into a sequence of acoustic feature vectors carrying features of the signal. And this is known as pre-processing i.e., feature extraction is completed here. We Transform audio wave to the frequency domain using `audio to fft`

## Model definition

A usual CNN architecture consists of a sequence of layers. Each layer transforms the input data by applying layer specific operations on the input and passing it over to the next layer. The three most common layer types found in a CNN architecture are: Convolutional Layer, Pooling Layer and Fully-Connected Layer. The convolutional layer in a CNN is where majority of the learning process takes place. Design and placement

of the filters along the various layers of a CNN determine the “concepts” that are learned at each layer.

We design one-dimensional convolutional neural networks (1D-CNN) with activation function softmax which learn speaker dependent features from the fft based speech representations for performing speaker recognition. The filter responses from each of the convolutional layers are made to pass through ReLU with activation function **softmax**. we used max-pooling to reduce the size of the parameter space to be learnt by the network then compile the network with Adam optimizer. Given that this is a 2-class problem, we use cross entropy loss

```
[ ] def residual_block(x, filters, conv_num=3, activation="relu"):
    # Shortcut
    s = keras.layers.Conv1D(filters, 1, padding="same")(x)
    for i in range(conv_num - 1):
        x = keras.layers.Conv1D(filters, 3, padding="same")(x)
        x = keras.layers.Activation(activation)(x)
    x = keras.layers.Conv1D(filters, 3, padding="same")(x)
    x = keras.layers.Add()([x, s])
    x = keras.layers.Activation(activation)(x)
    return keras.layers.MaxPool1D(pool_size=2, strides=2)(x)

def build_model(input_shape, num_classes):
    inputs = keras.layers.Input(shape=input_shape, name="input")

    x = residual_block(inputs, 16, 2)
    x = residual_block(x, 32, 2)
    x = residual_block(x, 64, 3)
    x = residual_block(x, 128, 3)
    x = residual_block(x, 128, 3)

    x = keras.layers.AveragePooling1D(pool_size=3, strides=3)(x)
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(256, activation="relu")(x)
    x = keras.layers.Dense(128, activation="relu")(x)

    outputs = keras.layers.Dense(num_classes, activation="softmax", name="output")(x)
```

## MODEL

## Training

```
▶ history = model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=valid_ds,
    callbacks=[earlystopping_cb, mdlcheckpoint_cb],
)
```

With 20 epochs and batch size 128 we got approximate **98%** accuracy

```

53/53 [=====] - 77s 1s/step - loss: 0.1401 - accuracy: 0.9455 - val_loss: 0.0961 - val_accuracy: 0.9667
Epoch 7/20
53/53 [=====] - 78s 1s/step - loss: 0.1285 - accuracy: 0.9535 - val_loss: 0.0920 - val_accuracy: 0.9747
Epoch 8/20
53/53 [=====] - 78s 1s/step - loss: 0.0936 - accuracy: 0.9640 - val_loss: 0.1200 - val_accuracy: 0.9680
Epoch 9/20
53/53 [=====] - 78s 1s/step - loss: 0.1200 - accuracy: 0.9569 - val_loss: 0.1100 - val_accuracy: 0.9600
Epoch 10/20
53/53 [=====] - 79s 1s/step - loss: 0.0922 - accuracy: 0.9649 - val_loss: 0.1078 - val_accuracy: 0.9680
Epoch 11/20
53/53 [=====] - 78s 1s/step - loss: 0.0776 - accuracy: 0.9711 - val_loss: 0.0886 - val_accuracy: 0.9667
Epoch 12/20
53/53 [=====] - 79s 1s/step - loss: 0.0727 - accuracy: 0.9720 - val_loss: 0.1005 - val_accuracy: 0.9693
Epoch 13/20
53/53 [=====] - 79s 1s/step - loss: 0.0722 - accuracy: 0.9744 - val_loss: 0.0862 - val_accuracy: 0.9733
Epoch 14/20
53/53 [=====] - 79s 1s/step - loss: 0.0574 - accuracy: 0.9787 - val_loss: 0.0618 - val_accuracy: 0.9840
Epoch 15/20
53/53 [=====] - 78s 1s/step - loss: 0.0552 - accuracy: 0.9804 - val_loss: 0.1152 - val_accuracy: 0.9707
Epoch 16/20
53/53 [=====] - 79s 1s/step - loss: 0.0569 - accuracy: 0.9803 - val_loss: 0.0708 - val_accuracy: 0.9827
Epoch 17/20
53/53 [=====] - 78s 1s/step - loss: 0.0468 - accuracy: 0.9816 - val_loss: 0.0647 - val_accuracy: 0.9853
Epoch 18/20
53/53 [=====] - 78s 1s/step - loss: 0.0574 - accuracy: 0.9800 - val_loss: 0.0778 - val_accuracy: 0.9787
Epoch 19/20
53/53 [=====] - 78s 1s/step - loss: 0.0438 - accuracy: 0.9830 - val_loss: 0.0889 - val_accuracy: 0.9667
Epoch 20/20
53/53 [=====] - 79s 1s/step - loss: 0.0461 - accuracy: 0.9828 - val_loss: 0.0666 - val_accuracy: 0.9813

```

## Evaluation of model

```
[ ] print(model.evaluate(valid_ds))s
```

```

24/24 [=====] - 8s 310ms/step - loss: 0.0666 - accuracy: 0.9813
[0.06662255525588989, 0.981333315372467]

```

**We got ~ 98% validation accuracy.**

We take some samples and:

**we can predict speaker speaker by uploading audio to  
gui by using trained model**

## Speaker Identification

Speaker Identification with MFCC &amp; SVM

Speaker Identification with FFT &amp; CNN

Select Audio

Play Audio

You have selected an Audio of Benjamin\_Netanyau

Click here to plot the signal :

Plot

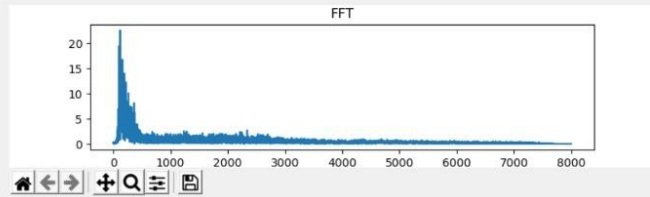
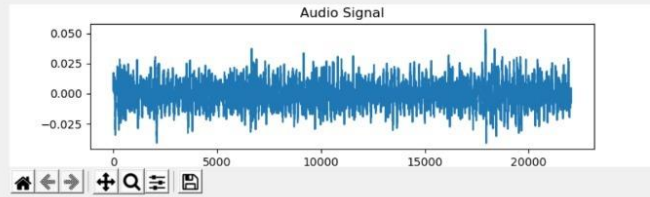
Create features and plot features :

MFCC/FFT

Predict using CNN

Predicted speaker is (CNN):

Benjamin\_Netanyau



**Thank you**