A project on

AUDIO CLASSIFICATION USING PYTHON

Submitted on partial fulfillment of award of

Bachelor in Technology

In

Computer Science and Engineering

By

Nani Tunia (D/18/CS/119)

Priyanshi Borah (D/18/CS/104)

Under the Supervision of

Mr. Kapang Legoh

Associate Professor

(Department of Computer Science and Engineering)

Department of Computer Science and Engineering

North Eastern Regional Institute of Science and Technology

Deemed-To-Be-University under the ministry of education, Govt. of India

Jan-May, 2021-22

DECLARATION

We hereby declare that the work which is being presented in this report entitled "Audio classification", a prerequisite towards partial fulfillment for the award of deegree in Computer science and Engineering Department, North Eastern Regional Institute of Science and Technology(Deemed University), Nirjuli, Arunachal Pradesh, 791109, is an accurate record of our project work carried out under the guidance of Mr Kapang Legoh, Computer Science and Engineering Department. This work contains no material which has been used for the award of some other degree in other university.

Nani Tunia	Priyanshi Borah
(D/18/CS/119)	(D/18/CS/104)

CERTIFICATE OF APPROVAL

Certified that the project report entitled "**AUDIO CLASSIFICATION ON CAT'S AND DOG'S "** is a bonafide work carried out jointly by **Nani Tunia (D/18/CSE/119) and Priyanshi Borah (D/18/CSE/104).** The project report embodies the original work done by them towards partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** at **North EasternRegional Institute of Science and Technology, Arunachal Pradesh**. It is understood by this approval that the undersigned do not endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project report only for the purpose for which it has been submitted.

Mr. Kapang Legoh
(Project Supervisor)
Asst. Professor
Department of Computer Science & Engineering

Mr.Aswini kumar Patra(ProjectCoordinator)Asst. Professor
Department of Computer Science & Engineering

Dr. Moirangthem Marjit Singh(Head of the Department)

Department of Computer Science & Engineering

ACKNOWLEDGEMENT

No task can be achieved alone, particularly while attempting to finish a project of this magnitude. It took many people to facilitate it and support it. Hence we would like to acknowledge all of their value valuable support and convey our humble gratitude to them.

We would like to thank our project guide Mr Kapang Legoh(Assist. Professor) and to our project co-ordinator Mr Aswini Kumar Patra(Assist. Professor) of computer science and engineering department, NERIST, who has always been open to discussion and frequently enquired about the project and any problem faced. He has also given us valuable guidance as how to go about the project.

We have put our best effort to make this project usable, informative, and understandable as possible. We have done the best we could do and have been honest to the professor and most importantly to ourselves.

Table of Contents

Table of figures:

ABSTRACT

Our project illustrates a straightforward audio classification model supported deep learning. we tend to address the matter of classifying the sort of sound supported short audio signals and their generated spectrograms, from classifying dog's audio to cat's audio throughout model ttraining. So as to satisfy this challenge, we tend to use a model supported Convolutional Neural Network (CNN). The audio was processed with Mel-frequency Cepstral Coefficients (MFCC) into what is unremarkably called Mel spectrograms, and hence, was reworked into a picture. Our final CNN model achieved 89% accuracy on the testing dataset.

Chapter 1. Introduction

1.1 Project Overview :

The input to our model, in this project, is cats associated dogs recording an audio go in WAV kind. It lies below the supervised machine learning class. Thus, a dataset is also present as well as a target class. Hence, the intention here is to classify if the given input wav file is that of a cat or dog. Each of the dog and cat sound is incredibly distinguished like in their pitch and frequency level since completely different| sounds have different sample rates. By default, Librosa mixes all audio to mono and resample's them to 22050 cycles/second at load time. For music and audio analysis, Librosa is associate ASCII text file python package. The info and the sampling rate are provided by Librosa. Audio or sound, is in its raw kind, that the data provided should be pre- processed to extract significant and meaningful features so we implemented an algorithm i.e., MFCC (Mel Frequency Cepstral Coefficients) rule. Then, when audio extraction is done, the information is fed and the dataset is split in training and test set. So, after the preprocessing, a Convolutional Neural Network model is designed using tensor flow. For every code and model building, Keras API was used to implement Google colab.

1.2 Motivation

Machine learning can be used in image processing, understanding speech, and musical instruments, speech to text, environmental sound classification and many more. And as for our project we implemented a class of speech processing i.e, audio classification. Converting sound waves into audio and spectrograms which is a visual representation of frequencies with the help of function provided by the machine learning.

There are many techniques to classify images as many different in-built neural networks under CNN are already there, especially if it is related to images. And it's straightforward to extract options from pictures as a result of pictures already are available the shape of numbers, because the formation of a picture may be a assortment of pixels, and pixels area unit within the sort of numbers. When we have data as text, we use the sequential encoder and decoder-based techniques to find features. But if it is to sound recognition or audio it is more difficult compare to text because it is based on frequency and time. Therfore a proper model is to be made to extract the frequency and pitch of that of audio so as to make it easier to later recognize it.

Chapter 2. Preliminaries and background

2.1 Related work

Machine learning: Image classification of cats and dogs – Before a decade, in computer notion, many problems had been saturating in accordance on their precision. However, the accuracy of those troubles significantly stepped forward with the boom of deep gaining knowledge of strategies. The majority of the problems that arises of image class is that it is defined as predicting the distinct categories an photo can belong to. Hence, for the supplied enter/ photograph detection with the aim of accomplishing a high precision, a state of the art approach is incorporated, i.e., a convolutional neural nnetwork turned into build for the photo category mission of puppies and cats. A dataset become given from Kaggle comprising of a total 25000 pix of each dogs and cats.

Machine learning: Audio classification of different bird species - Here, the methodology and results on using deep learning to assist in classification of birds by their sounds are presented. As birds indicate the health of an ecosystem, hence this topic is of high importance. Random Forest Classification and custom-made six CNN models from literature was performed on a dataset of ten birds that were composed from xeno-canto.org. The highest accuracy was achieved at around 65% by the Random Forest and at about 58% for CNN model.

Chapter 3. Method

3.1 Dataset

Many online resources exist for dog and cat audio dataset. We downloaded our dataset from Kaggle. The datasets consists in many “wav” files for both cat and dogs classes.

- Cat has 164 WAV files to which corresponds 1323 sec of audio.
- dog has 113 WAV files to which corresponds 598 sec of audio.

Datasets	**Train size**	**Test size**
Cats	106	58
Dogs	79	34

Table 1: Dataset statistics

The dataset in keras is divided into folders for each class - training and testing set. So for the given training set and the given test set, there are two folders which is composed of, one for cat audio and the other for the dog audio. All the WAV files contains 16KHz audio and have variable length.

3.2 Feature extraction

Since the project is on a supervised learning task; conjointly the input are going to be audio of cat and dog and there is a need to predict if it's either of them. The raw audio signal cannot be taken as input to our model, as there will be heaps of noise within the audio signal. Then the raw audio signal is extracted and used as spectrogram input to the bottom model. This offers a better overall performance than thinking about the raw audio signal as enter without delay. Thus, MFCC is implemented, a extensively used technique for extracting the options from the audio sign. Following figure is that the model for MFCC.

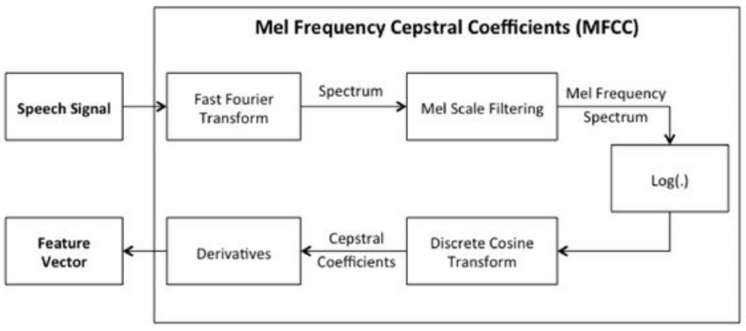


Fig 3.1. block diagram for MFCC model

Mel Frequency Cepstral Coefficients also coined as MFCCs are a feature that is widely used in various neural network related work or deep learning such as automatic speech and speaker recognition. This technique is used for feature extraction includes various steps. These steps include the following

1. Window the given signal
2. DFT implementation
3. Take the log of the magnitude
4. Wrapping the frequencies on a Mel scale which after do the inverse of DCT.

For sound processing, mel – frequency cepstrum which is portrayal for short

term power spectrum of a sound is actually based on linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.
Deprived from a type of cepstral depiction of the audio clip namely a non linear "spectrum of a spectrum", MFC is made by collectively MFCC coefficient. The difference between the mel frequency spectrum and just cepstrum is that: In MFC, the frequency bands are equally spaced on the mel scale, which is approximate the human auditory system's response more closely than the linearly spaced frequency bands used in the normal spectrum. As a result, this frequency wrapping gives better representation of sound. MFCC is a algorithm that is mainly used to extract high frequency and low frequency data of a given input.

Implementation

Because of the input signal which is audio that changes repeatedly, so in order to clarify more of these things therefore we take the given audio signal statistically (statistically stationary) does not change, scales that are on short time. For which the signal that has to be framed is now segmented into 20-40 frames. As a result of it, if the frame is far shorter then not enough samples will be there to urge a reliable spectral estimate and the signal changes an excessive amount of throughout the frame if it's not longer. In the further step, the ability spectrum of every frame is calculated. The human tube-shaped structure which is a biological organ within the human ear that trembles at totally diverse areas of that portion where the frequency of the sounds that is taken as an input is focused, which motivates the whole concept of it. Totally different nerves hearth informing the brain that sure frequencies area unit gift which is observed by looking on the placement within the tube-shaped structure that vibrates (which wobbles little hairs). The periodogram spectral estimate still have plenty of noise or things that don't seem to be needed for the speech process and the result is additionally intense because the frequency will increase.

So as to total them up to urge an inspiration of what quantity energy exists in varied frequency regions, a clump of periodogram area unit is taken. Mel filterbank often performs this. The primary filter is extremely slender and provides a sign of what quantity energy exists close to zero Hertz. It is tended to dwindle involved regarding variations because as the trembling of the incoming sounds gets on its peak the filter which we are accumulating keeps getting wider. And the way to area our filter banks and the way wide to create them is told specifically to us by the Mel Scale. The index permits to use cepstral mean subtraction, which may be a channel normalization technique once filterbank energies is found, by taking the index of them.

To figure the DCT of the log filter bank energies is the final step. This step is completed as a result of filterbanks which is all overlapping and because of the relation between the filterbank energies area unit with each other. Energies are correlated by the DTC that refers to the modeling of features, which can be done by using diagonal convenience matrices.

3.2.1 Frame Blocking

The first step to operate the MFCC technique is frame blocking. Frame blocking is said to be the process of cutting a given sound signal into several frame. Here, in a given speech signal which is continuous is divided firstly into P samples. Firstly here we will consider the first frame as P samples and the second frames which begins with Q samples that also overlaps P-Q samples taken.

This method will continue till all the speech is considered within that intervals one frame or a lot of frames. The typical values generally for P and Q are P = 256; which is equivalent to thirty millisecond windowing and provides the quick radix-2 FFT and Q = one hundred.

3.2.2 Windowing

Next , every individual frame within the process is to be windowed. therefore, the signal need to be prevented from discontinuing and so at the start and the finish of every frame it needs to be minimized. The main objective is so to reduce the deformation of the input spectral to taper the given signal to zero by mistreatment at the very start of the frame and the finish of every frame. For the tendency to outline the given window denoted w(m), $0 \le m \le M-1$, such that M is the variety of provided samples in every frame after which the results of the above process is:

$$y_1(\mathrm{m}) = x_1(\mathrm{m})\mathrm{w}(\mathrm{m}),\ 0 \le \mathrm{m} \le \mathrm{M}-1$$

3.2.3 Fast fourier transform

Here , following next process is Fast Fourier transform, it basically will convert from the time domain into the frequency domain by taking every frame that’s taken of M sample. To implement the discrete Fourier transform (DFT), the FFT is used which is a quick rule and is outlined on the given input set of M samples $\{x_n\}$, as follow:

$$X_l = \sum_{m=0}^{M-1} x_m e^{-p2lm/M} \quad ,\ \mathrm{l} = 0,1,2,\ldots \mathrm{M}-1$$

3.2.4 Mel-frequency wrapping

From psychophysical studies, it has been shown that a linear scale is not followed in human perception of the frequency contents of sound for speech signals. So a subjective pitch (referred to as the ‘mel’ scale) is measured on a scale for every tone with associate actual frequency, f, measured in cycle. The scale could be linear frequency spacing below one thousand cycles and a index spacing on top of one thousand cycle.

3.2.5 Cepstrum

Towards the end, now the log mel spectrum that is taken is then converted just like before. So the output we will get next is the mel frequency cepstrum coefficients also shorted as MFCC, that we get at the end. An honest illustration of the native spectral properties of the given frame that is computed is provided by the cepstral illustration the sound spectrum. And therefore we will now transform them back to the domain of the time mistreatment the distinct circular function rework also shorted as DCT, as a result of the mel spectrum coeeficients (their logarithm) square measure real numbers. So to calculate the MFCC $\widetilde{c_{\mathrm{m}}}$, as

$$\widetilde{c_m} = \sum_{k=1}^{k} \log(\widetilde{c_n}) \, cos[n(l - 1/2)\pi/l], \qquad \text{m=0,1…l-1}$$

Now a group of mel-frequency cepstrum coefficients is calculated out, by utilizing the series of steps that is given represented on top for every speech whose frame is of around 40msec with an additional overlap. Expressed on a mel-frequency scale, these square measure results of a circular function rework of the index of the short powered spectrum. It is termed as acoustic vector to these set of coefficients. An acoustic vectors which is in sequence is remodeled given input voices. So, in the upcoming section it will be seen that those acoustic vectors are often to acknowledge and represent the characteristic of the speaker’s voice.

3.3 Model building

Model is of various type that is used for various other performances. We are building a sequential model too create a deep neural networkst that constituent of stacks of layers and are connected in a sequence manner. By implementing the keras library, we implemented the modelling.

3.3.1 Introduction to CNN

Convolutional Neural Network is inspired by living beings and it is a type of deep learning model or multilyered neural network and it is formulated to mechanically and adjustively learn spacial hierarchies of options, from low to a very high level patterns. CNN could be a described as a mathematical ensemble that's usually cconsisted of 3 forms of layers : Convolution layer, Pooling layer, and totally connected layers. They are the part of hidden layer in the convolutional neural network. The first two layers that are convolution layers, pooling layers. Just like classification they performs the function of feature extraction, and the the third layer is called a completely connected layer basically what it does is it inputs the extracted value that was found or taken as input. In Convolutional Neural Network a convolution layer, consist of a layer of mathematical operations/ equation, that’s a specialised form of linear operation. Kernel is a small, consists of parameters with low grid in kernel the picturs or image element values area unit is kept in a very exceedingly two dimensional grids. Kernel is a favourable feature extractor that is applied at every picture/image position, that makes CNNs extremely affordable for image processing, since a feature might occur anyplace within the image. And naturally layer feeds its output into following layer, extracted options will now gradually become more and more advanced. The method of optimizing parameters like kernels is termed coaching, that is performed therefore on minimizing the distinction between outputs and ground truth labels through an improvement algorithm also which is also referred to as backpropagation and gradient descent.

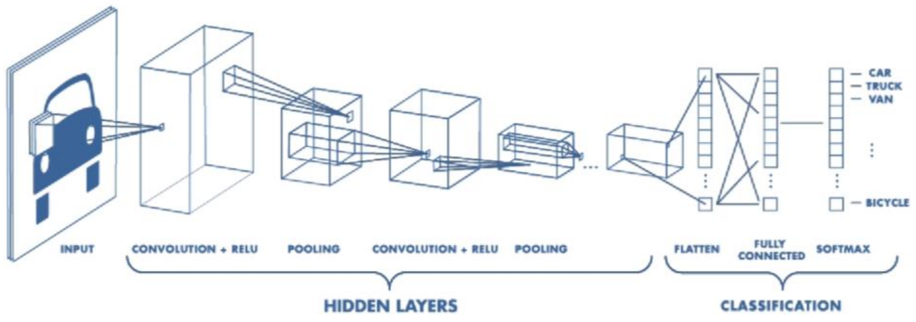


Fig 3.2. Convolutional Neural Network architecture

Convolution layer

Convolutional layer tends to learn about the feature structure of the inputs. Kernel is a set of learnable parameters a matrix, that will compute a mathematical operation on the two given matrices, and from which the alternative matrix is therefore taken as receptive field that is confined. A lot is going on inside a kernel’s structure than how we perceive it as, it looks spatially smaller which can be even compared as samller as an image. For example in a Red-blue-green channel of an image the information about the kernel like the height, width they are smaller when in spatial although since now that it is an RGB channel it will extend or can extend to any of the three channels. Throughout the channel, the kernel will always move towards the peak of the channel and on to the coverage the image have through out the channel that leads to the illustration of what is known as thereceptivee region. The process will now produce a output of the image which is a 2-D(two-dimensional) and which is also studied as an activation map. Activation map will now give feedback of the kernel at every special region of the given image that was taken as an input before. The summarized output can be calculated as below:

$$Wout = W - F + 2P / S + 1$$

F = spatial size ; S = stride

P = amount of padding P

Pooling Layer

After the convolution layer the next layer is the pooling layer in here it will now replace the output of the network that has just gone through the above layer at sspecified places by finding out a outline data point of the close output. By doing this it will now reduce the spatial size of the shown illustration that will again decrease the given quantity of weights and the computation. This layer is operated on each portion of the illustration one by one. Pooling functions are of many type like the following:

1. typical of the oblong neighborhood,
2. L2 norm of the oblong neighborhood,
3. Weighted average, which will support the central picture element that forms a gap

But with all this functions available the first method that is being used is scoop pooling, it provides with the most number of output that is achieved from the neighborhood.

$$W_{out} = \frac{W-F}{S} + 1$$

Formula for padding layer

Translation unchangeableness meaning that the degree of the object will be distinguishable even though it is in wherever portion of the channel/frame. And this characteristic is provided by the pooling layer.

Fully Connected Layer

In here after the pooling layer now the neurons will have whole full connection combining will all pre and post layer of the neurons just like a fully connected layer. Fully connected layer can be evaluated mathematically by multiplying a matrix and then after adding a bias.The main feature of the fully connected layer is that it portrays the relation among the output and the given input.

Non-Linearity Layers

In non-linearity layer it is conducted when the convolution layer has now presented or introduced nonlinearity to the given activation map. It usually is computed after the nonlinearity introduced by the first layer. Also since we know that convolution layer is a linear function or the one that operates on linearity the extracted image from it are not even close to linear so non linearity layer is a must. There are many varieties of non-linear operations such as Tanh, Sigmoid, Max-out, SoftMax activation functions. We have intendent to enforce ReLu(Rectified Linear Unit) activation function for our model.

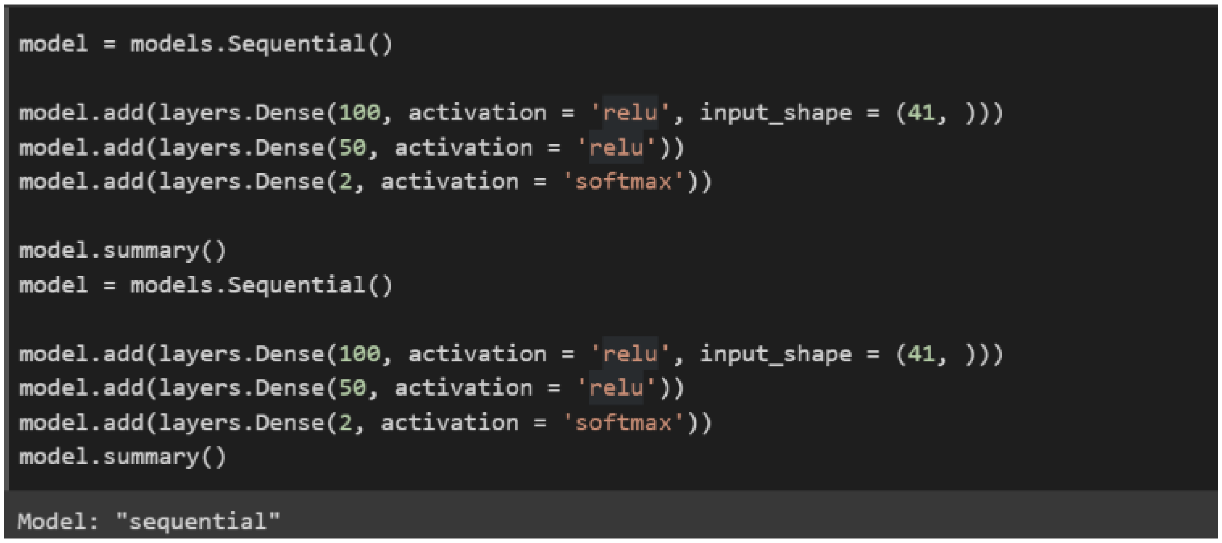
ReLU

It is mathemathically represented as: f(x) =max (0,x) which means, that activation function will be threshold at zero i.e if the value is negative it is converged to 0 and if positive then 1. Comparatively to sigmoid and tanh, ReLU is very much reliable and it accelerates the convergence of the value by sixfold. But the disadvantage is that ReLU will be very fragile throughout coaching. Neuron is updated in such a way that if an value passing through will never get more updated again.

3.3.2 Designing the Convolutional Neural Network :

We designed our Convolutional Neural Network such that there was a 2-layer deep architecture that consisted of final fully connected layer and a output prediction layer.

The code formulated of the above implementation method is shown in Fig. 3.3.



```
model = models.Sequential()

model.add(layers.Dense(100, activation = 'relu', input_shape = (41, )))
model.add(layers.Dense(50, activation = 'relu'))
model.add(layers.Dense(2, activation = 'softmax'))

model.summary()
model = models.Sequential()

model.add(layers.Dense(100, activation = 'relu', input_shape = (41, )))
model.add(layers.Dense(50, activation = 'relu'))
model.add(layers.Dense(2, activation = 'softmax'))
model.summary()

Model: "sequential"
```

Fig 3.3. Code implementation

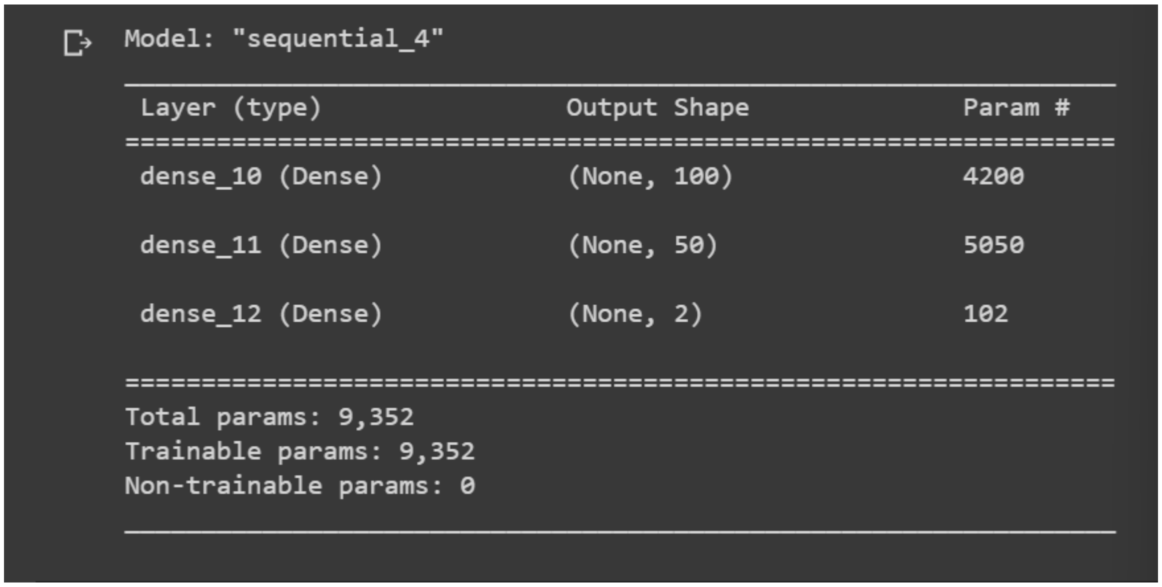
As from the above Fig 3.3, we will add the following layers in sequence step by step.

•Flatten layer: In this layer, it will now transform the given input into one-dimensional array. This layer doesn't consist any parameters and the main purpose of it is to do to do some preprocessing. Our input that we gave for the flatten layer is of form i.e inputshape=(41,).

• Dense hidden layer: It is the second layer which has like fifty neuron. And for outlining the 2nd layer,i.e the dense layer we used ReLU activation operation.

• And now in the final layer we tend to modification of the dense output layer with two neurons employing the softmax activation operate, resulting the completely different categories.

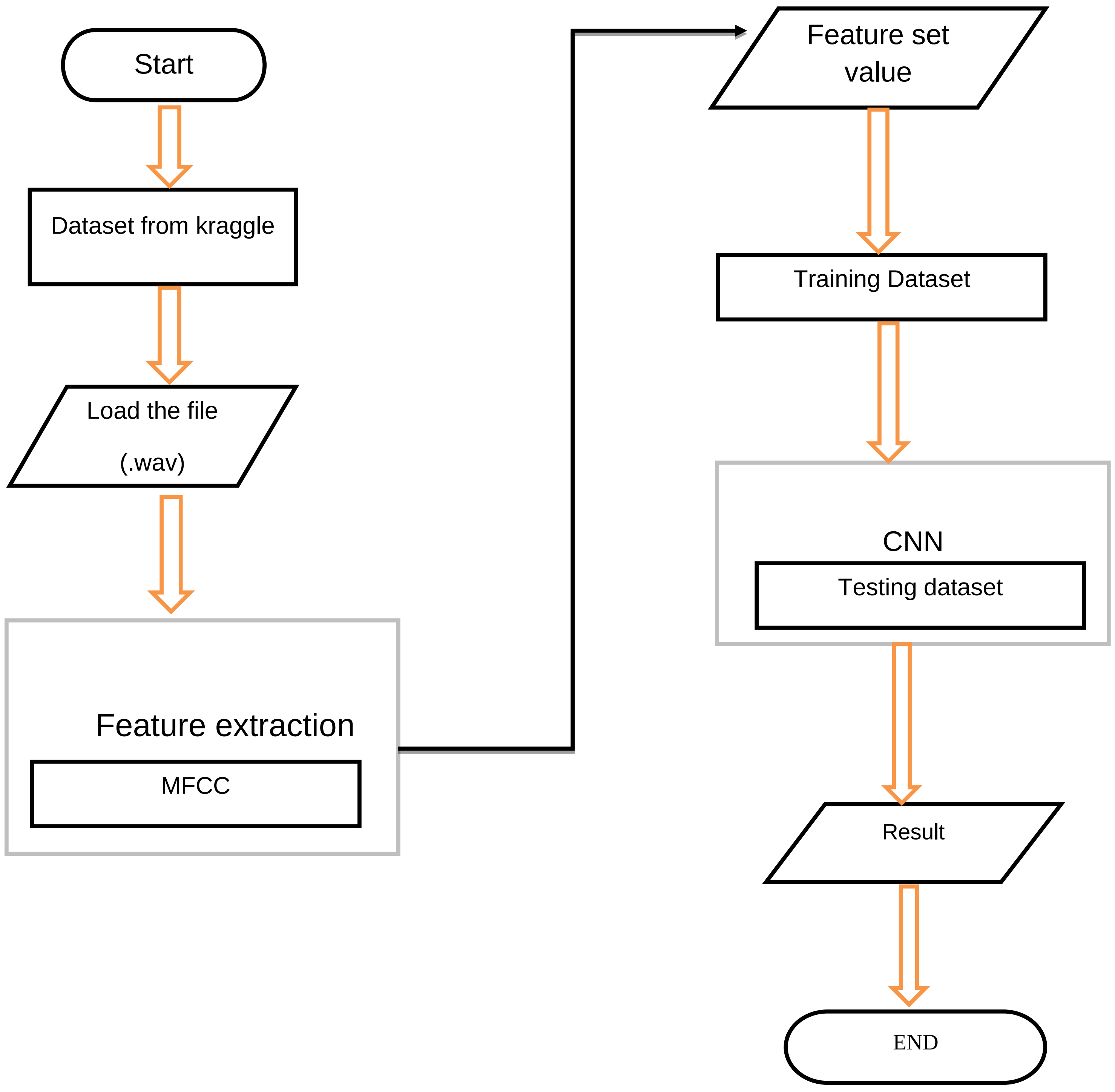
The network architecture of the above is given below in Fig3.4:



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 100)               4200

 dense_11 (Dense)            (None, 50)                5050

 dense_12 (Dense)            (None, 2)                 102

=================================================================
Total params: 9,352
Trainable params: 9,352
Non-trainable params: 0
_________________________________________________________________
```

Fig 3.4. Network Architecture output

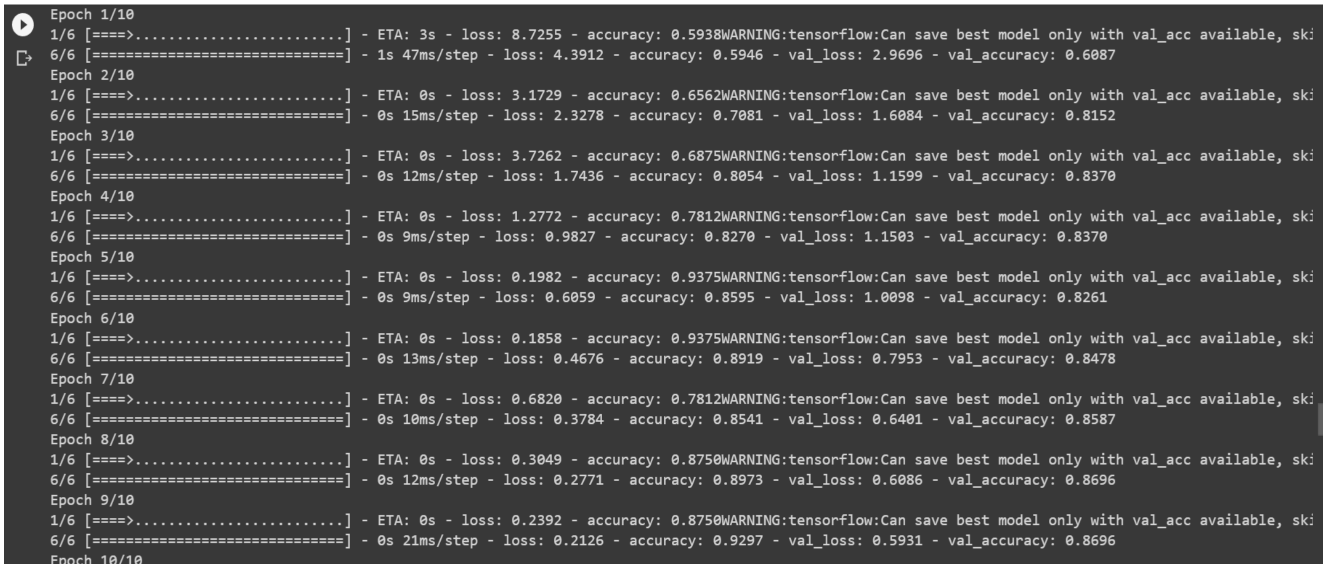
3.3.3 Flowchart:



Chapter 4.   Experiment and results

4.1 Training and testing

Training our model until 10 epochs:



```
Epoch 1/10
1/6 [====>.........................] - ETA: 3s - loss: 8.7255 - accuracy: 0.5938WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 1s 47ms/step - loss: 4.3912 - accuracy: 0.5946 - val_loss: 2.9696 - val_accuracy: 0.6087
Epoch 2/10
1/6 [====>.........................] - ETA: 0s - loss: 3.1729 - accuracy: 0.6562WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 15ms/step - loss: 2.3278 - accuracy: 0.7081 - val_loss: 1.6084 - val_accuracy: 0.8152
Epoch 3/10
1/6 [====>.........................] - ETA: 0s - loss: 3.7262 - accuracy: 0.6875WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 12ms/step - loss: 1.7436 - accuracy: 0.8054 - val_loss: 1.1599 - val_accuracy: 0.8370
Epoch 4/10
1/6 [====>.........................] - ETA: 0s - loss: 1.2772 - accuracy: 0.7812WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 9ms/step - loss: 0.9827 - accuracy: 0.8270 - val_loss: 1.1503 - val_accuracy: 0.8370
Epoch 5/10
1/6 [====>.........................] - ETA: 0s - loss: 0.1982 - accuracy: 0.9375WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 9ms/step - loss: 0.6059 - accuracy: 0.8595 - val_loss: 1.0098 - val_accuracy: 0.8261
Epoch 6/10
1/6 [====>.........................] - ETA: 0s - loss: 0.1858 - accuracy: 0.9375WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 13ms/step - loss: 0.4676 - accuracy: 0.8919 - val_loss: 0.7953 - val_accuracy: 0.8478
Epoch 7/10
1/6 [====>.........................] - ETA: 0s - loss: 0.6820 - accuracy: 0.7812WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 10ms/step - loss: 0.3784 - accuracy: 0.8541 - val_loss: 0.6401 - val_accuracy: 0.8587
Epoch 8/10
1/6 [====>.........................] - ETA: 0s - loss: 0.3049 - accuracy: 0.8750WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 12ms/step - loss: 0.2771 - accuracy: 0.8973 - val_loss: 0.6086 - val_accuracy: 0.8696
Epoch 9/10
1/6 [====>.........................] - ETA: 0s - loss: 0.2392 - accuracy: 0.8750WARNING:tensorflow:Can save best model only with val_acc available, ski
6/6 [==============================] - 0s 21ms/step - loss: 0.2126 - accuracy: 0.9297 - val_loss: 0.5931 - val_accuracy: 0.8696
Epoch 10/10
```

Fig 4.1. Epoch training

Here the accuracy is 85%.s

From fig.4.2 the graph shows the training and validation accuracy using Relu function. As we can observe from the figure that the training accuracy(blue line) is increasing while using 10 epochs. The accuracy can be improved by taking more than 10 epochs. But for our model 10 epoch worked fine!

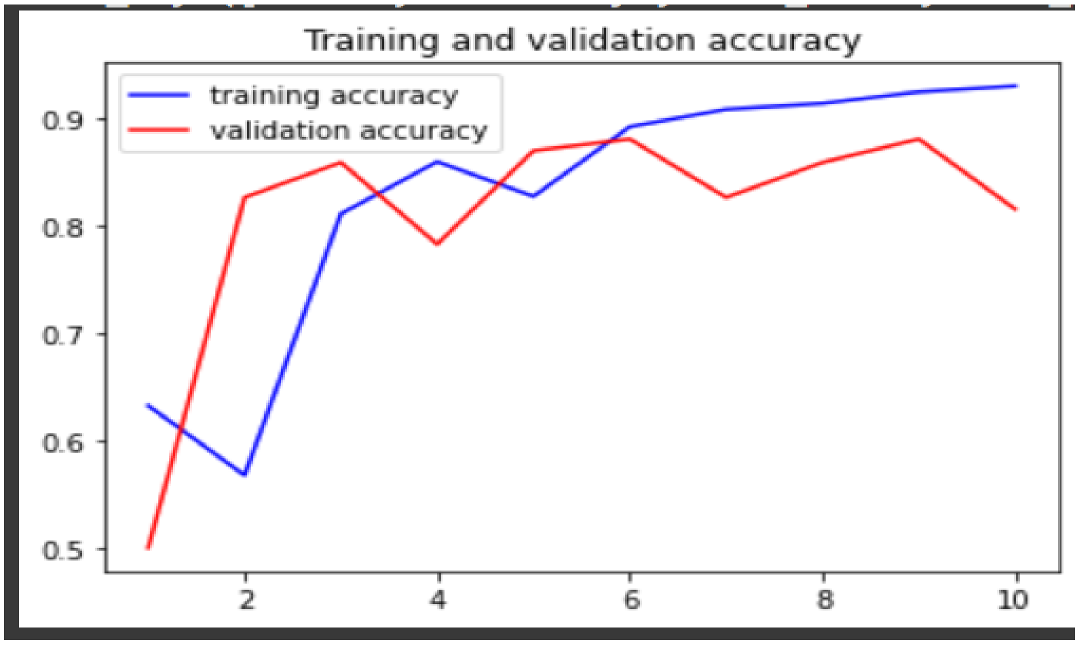


Fig 4.2. Training and accuracy graph

The following graph below fig.4.3 shows the graph for random audio input for example the axs[0][0] and axs [1][0] is of cat and axs[0][1] & axs[1][1] is of dog.

The audio doesn’t just constitute that of cat and dog it comes along with background noises and etcetera.

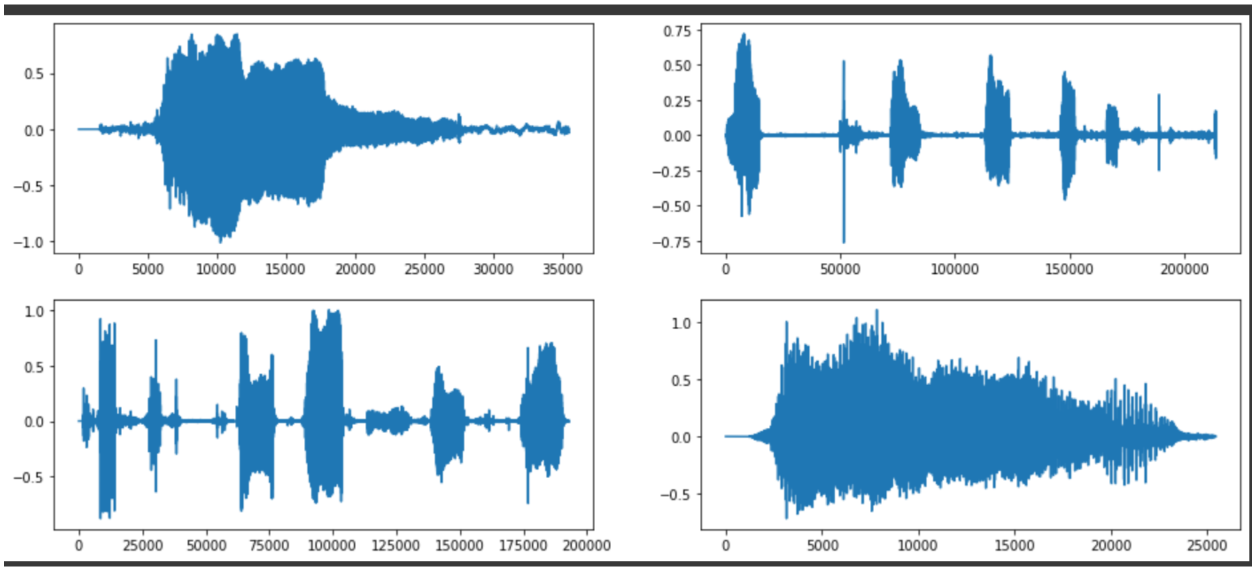
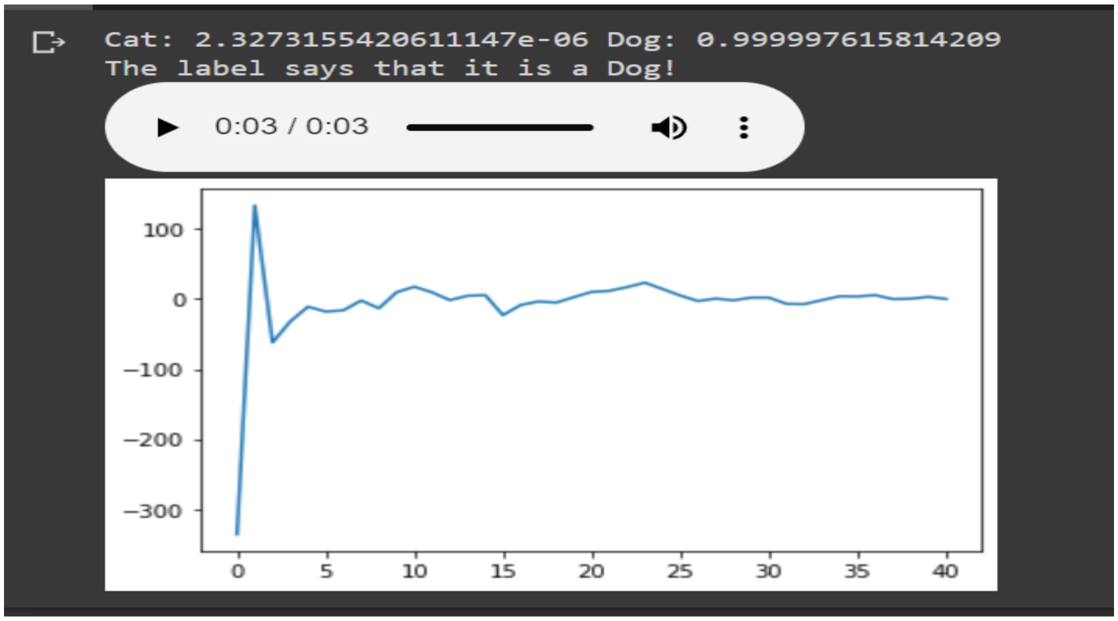


Fig 4.3.random input graph

Now after testing the model with any random input we got the following as shown in the fig.4.4:



```
Cat: 2.3273155420611147e-06 Dog: 0.999997615814209
The label says that it is a Dog!
```

Fig 4.4. output of an example

From the above figure we observed that the accuracy that it’s a dog sound is 99% .

Chapter 5. Coding

#Initializing and importing libraries:

```
from os import listdir

from os.path import isfile, join

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import pandas as pd

import numpy as np

import pylab as pl

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from sklearn.ensemble import (RandomForestClassifier, ExtraTreesClassifier,

                              AdaBoostClassifier)

from sklearn.svm import SVC

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# Any results you write to the current directory are saved as output.

import os

sounds = os.listdir("/content/cats_dogs")

print(os.listdir("/content/cats_dogs"))

import warnings

warnings.filterwarnings("ignore")
```

#splitting the cats and dogs:

```
import glob

import ntpath

# List the wav files

ROOT_DIR_TEST = glob.glob('/content/cats_dogs/test')[0]

ROOT_DIR_TRAIN = glob.glob('/content/cats_dogs/train')[0]



X_path = glob.glob(ROOT_DIR_TEST + "/test/*") # test = dogs in this case !
(wrong name of directory was given when it was created)

X_path = X_path + glob.glob(ROOT_DIR_TEST + "/cats/*")

X_path = X_path + glob.glob(ROOT_DIR_TRAIN + "/dog/*")

X_path = X_path + glob.glob(ROOT_DIR_TRAIN + "/cat/*")

print (len(X_path))
```

# Split train and test:

```
X_train, X_test, y_train, y_test = train_test_split(X_path, y, test_size=0.33)


print("in X, there is {} cats and {} dogs".format(len(y) - sum(y), sum(y)))

print("in X_train, there is {} cats and {} dogs".format(len(y_train) - sum(y_train),
sum(y_train)))

print("in X_test, there is {} cats and {} dogs".format(len(y_test) - sum(y_test),
sum(y_test)))
```

```
print("len(y_train)",len(y_train))
print("len(y_test)",len(y_test))
print(y_test.shape)
print(y_train.shape)
```

#output:

```
in X, there is [164.] cats and [113.] dogs
in X_train, there is [113.] cats and [72.] dogs
in X_test, there is [51.] cats and [41.] dogs
len(y_train) 185
len(y_test) 92
(92, 1)
(185, 1)
```

# function to extract all the features needed for the classification

```
def extract_features(audio_samples, sample_rate):
    extracted_features = np.empty((0, 41, ))
    if not isinstance(audio_samples, list):
        audio_samples = [audio_samples]

    for sample in audio_samples:
        # calculate the zero-crossing feature
        zero_cross_feat = librosa.feature.zero_crossing_rate(sample).mean()

        # calculate the mfccs features
        mfccs = librosa.feature.mfcc(y=sample, sr=sample_rate, n_mfcc=40)
        mfccsscaled = np.mean(mfccs.T,axis=0)

        # add zero crossing feature to the feature list
        mfccsscaled = np.append(mfccsscaled, zero_cross_feat)
        mfccsscaled = mfccsscaled.reshape(1, 41, )

        extracted_features = np.vstack((extracted_features, mfccsscaled))

    # return the extracted features:
    return extracted_features

#import libraries for model creation:
from keras import layers
from keras import models
from keras import optimizers
from keras import losses
from keras.callbacks import ModelCheckpoint,EarlyStopping
#Convert the labels to match what our model will expect
from tensorflow.keras.utils import to_categorical
train_labels = to_categorical(y_train)
test_labels = to_categorical(y_test)

#Choose the parameters to train the neural network
best_model_weights = './base.model'
checkpoint = ModelCheckpoint(
    best_model_weights,
    monitor='val_acc',
    verbose=1,
    save_best_only=True,
    mode='min',
    save_weights_only=False,
    period=1
)

callbacks = [checkpoint]

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

#to train the model:

```
history = model.fit(
    X_train_features,
    train_labels,
    validation_data=(X_test_features,test_labels),
    epochs = 10,
    verbose = 1,
    callbacks=callbacks,
)
```

#Save the model

```
model.save_weights('model_wieghts.h5')
model.save('model_keras.h5')

# list all data in history
print(history.history.keys())

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'b', label = "training accuracy")
plt.plot(epochs, val_acc, 'r', label = "validation accuracy")
plt.title('Training and validation accuracy')
plt.legend()

plt.show()

#to try out an audio to compute the accuracy:
nr_to_predict = 50
pred = model.predict(X_test_features[nr_to_predict].reshape(1, 41,))

print("Cat: {} Dog: {}".format(pred[0][0], pred[0][1]))

if (y_test[nr_to_predict] == 0):
    print ("The label says that it is a Cat!")
else:
    print ("The label says that it is a Dog!")
```

```
plt.plot(X_test_features[nr_to_predict])
ipd.Audio(X_test[nr_to_predict],  rate=wav_rate)
```

Chapter 6. conclusion and future work

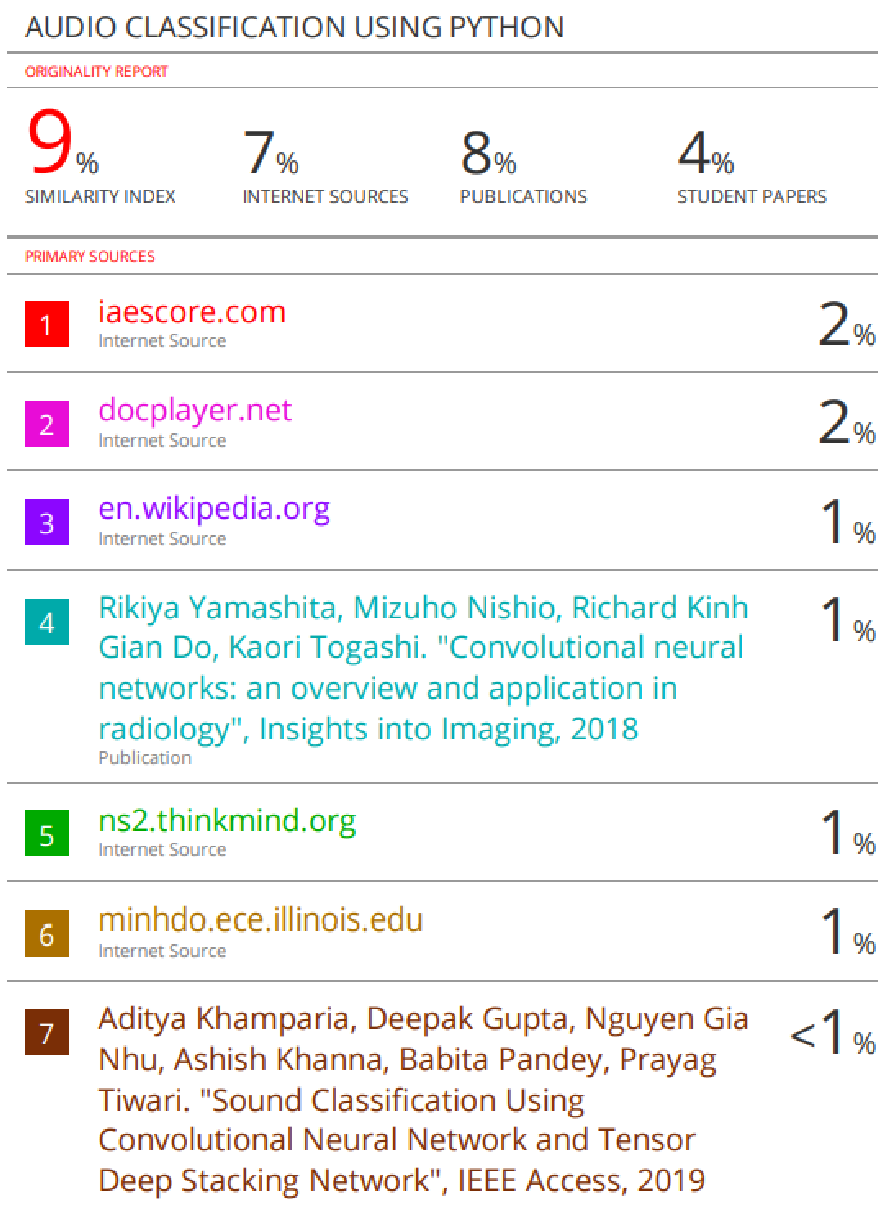
In this report, we first briefly explained the overview of this project and showed some referred project work alredy established. Then, we precisely illustrated our task, including the learning task and the performance task. After that, we explained the approach we are heading toward to inorder to classify the datasets. The approach/model we used is neural network which is a implementation of deep network which is trainable model by which we were able to classify the dogs and cats audio. The highest accuracy we got was 89.6%.

1. In the future we will try implement different high level model in order to achieve much higher accuracy.
2. We'll build a system that can directly intake a live raw audio.

Chapter 7. References

1. https://medium.com/@arcadius.modzelewski/3-ways-to-create-a-deep-neural-network-model-with-keras-and-tensorflow-2-0-659be9ce085d
2. https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html
3. https://www.section.io/engineering-education/machine-learning-for-audio-classification/
4. http://noiselab.ucsd.edu/ECE228-2020/projects/Report/48Report.pdf
5. https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/SINDURANSIVARAJAN.pdf
6. https://www.analyticsvidhya.com/blog/2022/03/implementing-audio-classification-project-using-deep-learning/

Plagiarism report:



AUDIO CLASSIFICATION USING PYTHON

ORIGINALITY REPORT

9%	7%	8%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	iaescore.com Internet Source	2%
2	docplayer.net Internet Source	2%
3	en.wikipedia.org Internet Source	1%
4	Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi. "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, 2018 Publication	1%
5	ns2.thinkmind.org Internet Source	1%
6	minhdo.ece.illinois.edu Internet Source	1%
7	Aditya Khamparia, Deepak Gupta, Nguyen Gia Nhu, Ashish Khanna, Babita Pandey, Prayag Tiwari. "Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network", IEEE Access, 2019	<1%