Cyber Gyan Virtual Internship Program

Centre for Development of Advanced Computing (CDAC), Noida

Intern: sreenandhana m

July-August 2024

Project Overview: Digital Audio Forensics

This project focuses on analysing audio to match the suspect's voice with the one from the sample files. The goal is to determine if the voice in the call recordings provided from the suspect’s device matches the voice recorded in the lab.

Problem Statement

We have a voice sample recorded in the lab from the accused. There are also three call recordings from the accused’s phone, believed to contain conversations between the accused and his mother. These conversations are suspected to include discussions about the crime. The task is to analyse if the voice in these call recordings is the same as the one recorded in the lab.

Technology and Tools

Tools and Libraries:

Artificial neural networks

- Pandas, librosa, matplotlib, TensorFlow, Scikit-learn, Keras

Infrastructure:

- Jupyter Notebook

Introduction to Audio Forensics

Definition:

Audio forensics applies scientific methods to analyze and compare audio recordings for investigative purposes, such as solving crimes and ensuring justice.

Role in Investigations:

Audio forensics plays a vital role in verifying evidence, identifying suspects, and helping to establish timelines and events related to criminal activities.

Objective:

The primary goal is to accurately determine if a suspect's voice matches a crime scene audio sample, providing critical clarity in legal procedures.

Artificial Neural Networks (ANNs)

Definition:

Artificial neural networks are computational systems inspired by the human brain, designed for tasks like pattern recognition and data classification across various domains.

Key Components:

ANNs are made up of neurons organized in layers—input, hidden, and output. During training, the connections between these neurons are adjusted to reduce prediction errors.

Applications:

They are used for image recognition, natural language processing, and audio classification, demonstrating their power in recognizing complex patterns.

Collecting Data for Audio Classification

Data Sources:

Audio data can come from online repositories, field recordings, or synthesized sounds. It's essential to gather diverse and representative datasets.

Audio Formats:

Common formats like WAV, MP3, and FLAC have specific characteristics that influence how they are processed and analyzed.

Preprocessing:

This step involves reducing noise, normalizing audio, and segmenting data to ensure high-quality input, which helps the model perform better.

Labeling:

For supervised learning, it's crucial to accurately label the audio data. Proper labeling leads to better model training, while poor labeling can result in unreliable predictions.

Feature Extraction Techniques

Spectrograms:

Spectrograms show how audio frequencies change over time, which is crucial in distinguishing different types of audio.

Mel Frequency Cepstral Coefficients (MFCCs):

Widely used in speech processing, MFCCs are extracted from audio signals and are excellent for representing the important features of sound.

Time Domain Features:

Features like zero-crossing rates and energy levels help analyze the structure and behavior of audio signals for classification.

Feature Selection Importance:

Choosing the right features is critical as it directly affects the model's ability to learn patterns. Dimensionality reduction techniques can help improve performance.

Training Neural Networks on Audio Data

Training Process:

Audio data is fed into the model, weights are adjusted using backpropagation, and the model iterates to reduce errors in classifying unseen data.

Hyperparameter Tuning:

Optimizing learning rates, batch sizes, and other hyperparameters can significantly boost model accuracy.

Overfitting & Underfitting:

Balancing between overfitting (learning noise) and underfitting (failing to capture important patterns) is essential for effective training.

Validation Techniques:

Using techniques like k-fold cross-validation ensures the model is robust and performs well across various datasets before being deployed in real-world settings.

Evaluating Audio Classification Models

Accuracy:

Measures how often the model predicts correctly.

Precision:

Shows the proportion of true positive predictions out of all positive predictions made by the model.

Recall:

Measures the proportion of true positive predictions out of all actual positive instances.

F1-Score:

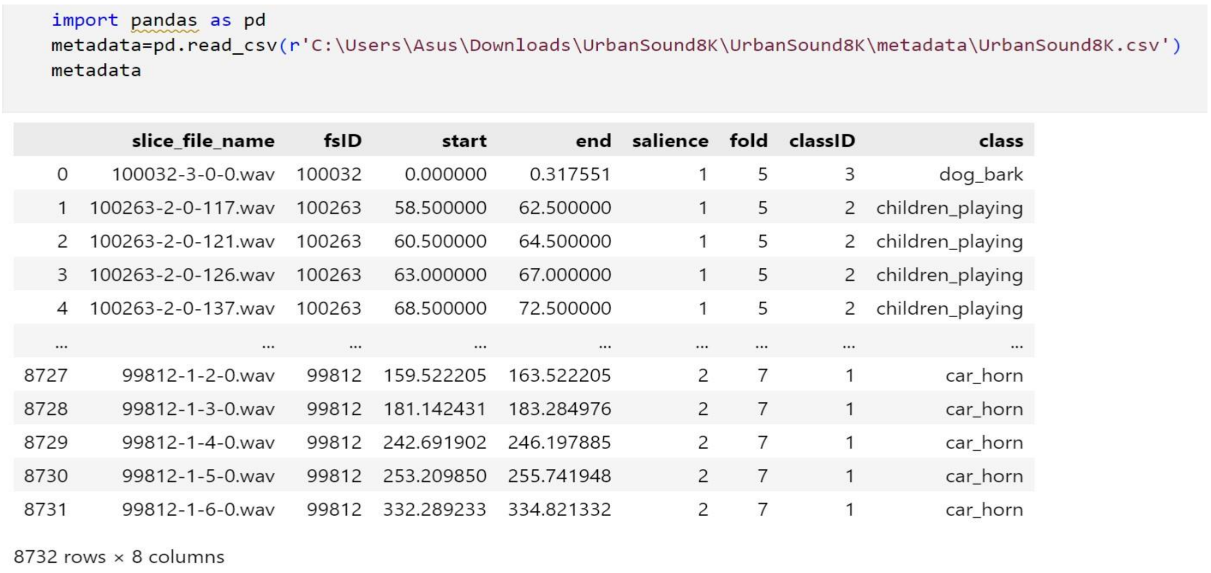
Balances precision and recall, which is particularly useful when working with imbalanced datasets.

IMPLEMENTATION:

The step-by-step process followed to solve the problem along with the screenshots of the steps taken

Gathering the data

•Creating metadata dataframe to summarise the audio files in the folder.

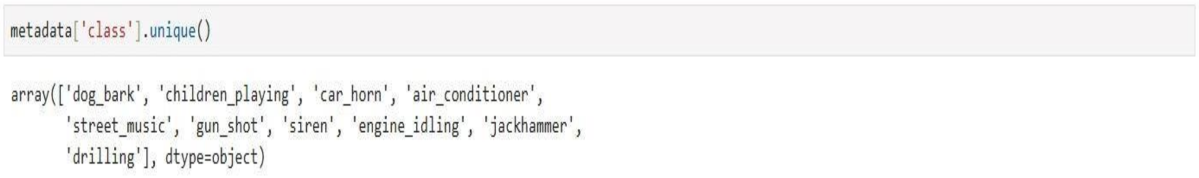


```
import pandas as pd
metadata=pd.read_csv(r'C:\Users\Asus\Downloads\UrbanSound8K\UrbanSound8K\metadata\UrbanSound8K.csv')
metadata
```

	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.000000	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.500000	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.500000	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.000000	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.500000	72.500000	1	5	2	children_playing
...	...	...	...	...	...	...	...	...
8727	99812-1-2-0.wav	99812	159.522205	163.522205	2	7	1	car_horn
8728	99812-1-3-0.wav	99812	181.142431	183.284976	2	7	1	car_horn
8729	99812-1-4-0.wav	99812	242.691902	246.197885	2	7	1	car_horn
8730	99812-1-5-0.wav	99812	253.209850	255.741948	2	7	1	car_horn
8731	99812-1-6-0.wav	99812	332.289233	334.821332	2	7	1	car_horn

8732 rows × 8 columns

•Getting know about the number of different classes present in the dataframe .



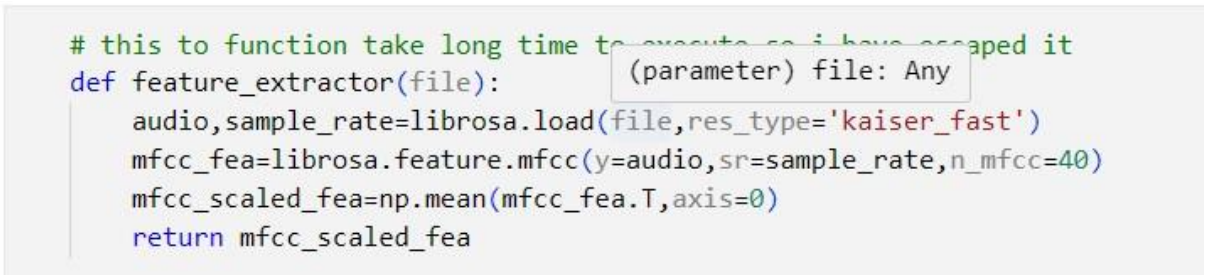
```
metadata['class'].unique()
```

```
array(['dog_bark', 'children_playing', 'car_horn', 'air_conditioner',
       'street_music', 'gun_shot', 'siren', 'engine_idling', 'jackhammer',
       'drilling'], dtype=object)
```

Now getting the feature(Mel-Frequency cepstral coefficient) of the each audio file present in the metadata using the librosa library

- **Creating the two functions : one to extract the feature of all the audio files and converting the features and the corresponding class in the dataframe**

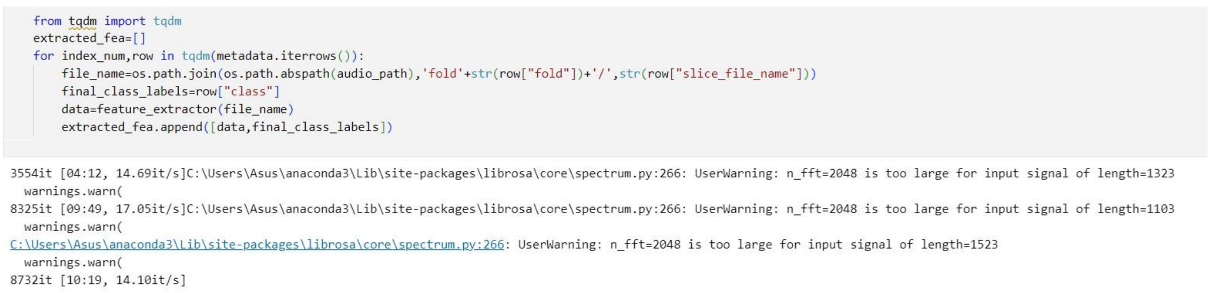
- **Function to extract the features of the audio file:**



```
# this to function take long time to execute so i have escaped it
def feature_extractor(file):
    audio,sample_rate=librosa.load(file,res_type='kaiser_fast')
    mfcc_fea=librosa.feature.mfcc(y=audio,sr=sample_rate,n_mfcc=40)
    mfcc_scaled_fea=np.mean(mfcc_fea.T,axis=0)
    return mfcc_scaled_fea
```

(parameter) file: Any

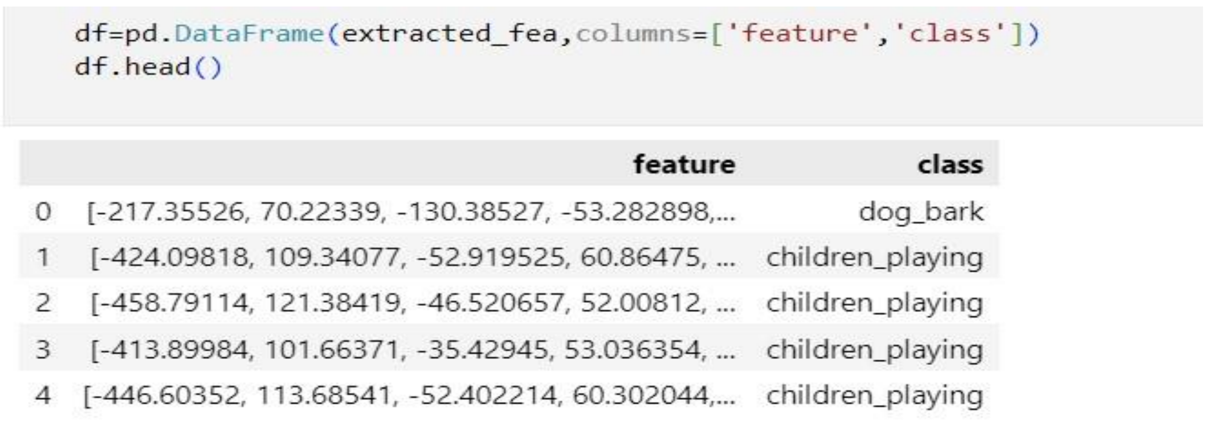
•Function to create the dataframe to the extracted feature and the classes



```
from tqdm import tqdm
extracted_fea=[]
for index_num,row in tqdm(metadata.iterrows()):
    file_name=os.path.join(os.path.abspath(audio_path),'fold'+str(row["fold"])+'/',str(row["slice_file_name"]))
    final_class_labels=row["class"]
    data=feature_extractor(file_name)
    extracted_fea.append([data,final_class_labels])
```

```
3554it [04:12, 14.69it/s]C:\Users\Asus\anaconda3\Lib\site-packages\librosa\core\spectrum.py:266: UserWarning: n_fft=2048 is too large for input signal of length=1323
  warnings.warn(
8325it [09:49, 17.05it/s]C:\Users\Asus\anaconda3\Lib\site-packages\librosa\core\spectrum.py:266: UserWarning: n_fft=2048 is too large for input signal of length=1103
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\librosa\core\spectrum.py:266: UserWarning: n_fft=2048 is too large for input signal of length=1523
  warnings.warn(
8732it [10:19, 14.10it/s]
```

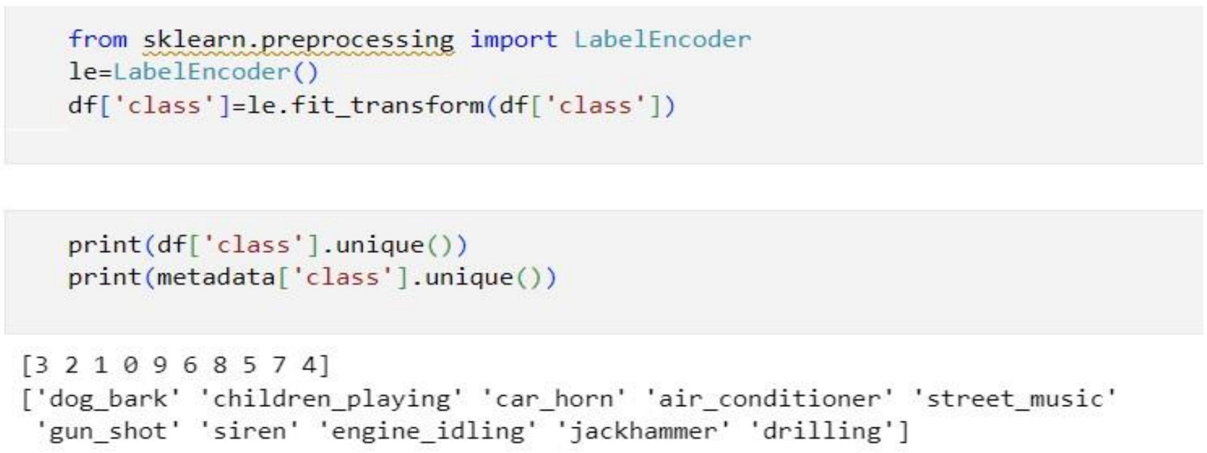
•The dataframe(df) is this:



```
df=pd.DataFrame(extracted_fea,columns=['feature','class'])
df.head()
```

	feature	class
0	[-217.35526, 70.22339, -130.38527, -53.282898,...	dog_bark
1	[-424.09818, 109.34077, -52.919525, 60.86475, ...	children_playing
2	[-458.79114, 121.38419, -46.520657, 52.00812, ...	children_playing
3	[-413.89984, 101.66371, -35.42945, 53.036354, ...	children_playing
4	[-446.60352, 113.68541, -52.402214, 60.302044,...	children_playing

•Encoding the classes of the sounds using the LabelEncoder()

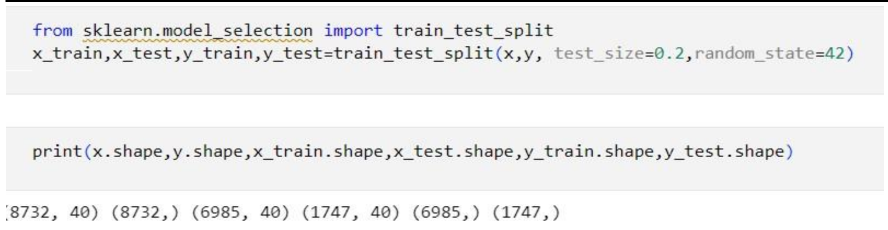


```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['class']=le.fit_transform(df['class'])
```

```
print(df['class'].unique())
print(metadata['class'].unique())
```

```
[3 2 1 0 9 6 8 5 7 4]
['dog_bark' 'children_playing' 'car_horn' 'air_conditioner' 'street_music'
 'gun_shot' 'siren' 'engine_idling' 'jackhammer' 'drilling']
```

•Splitting the data in the training and the testing dataframes

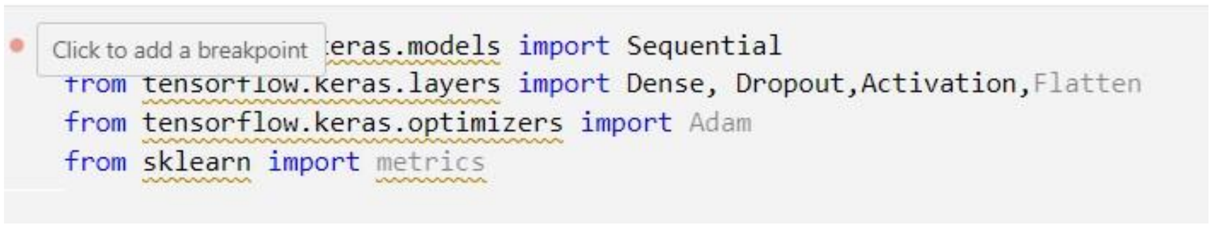


```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2,random_state=42)
```

```
print(x.shape,y.shape,x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(8732, 40) (8732,) (6985, 40) (1747, 40) (6985,) (1747,)
```

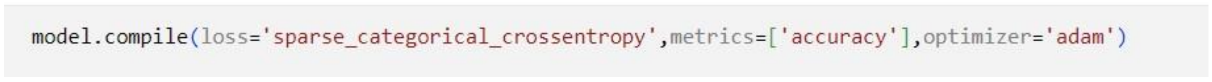
•Creating the Artificial Neural Network using the tensorflow and keras



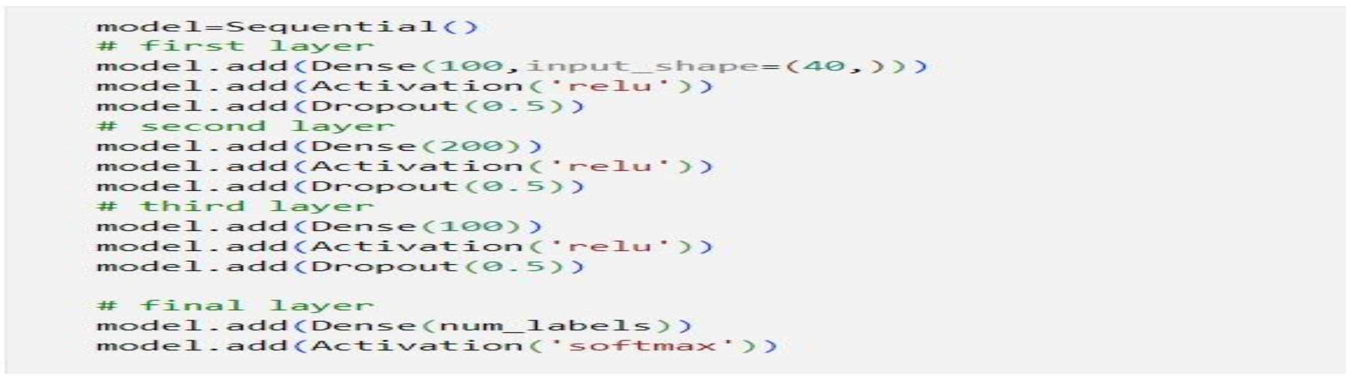
```
eras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,Activation,Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics
```

•Model is sequential type neural network

•Model summary(loss,optimizers):

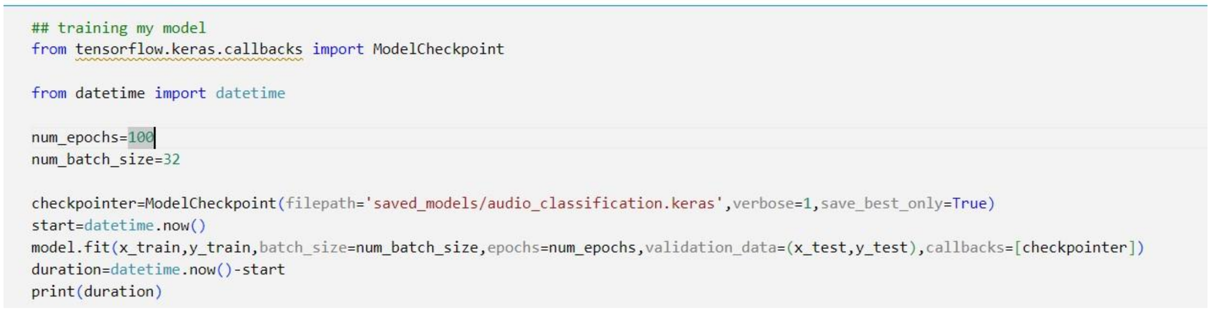


```
model.compile(loss='sparse_categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
```



```
model=Sequential()
# first layer
model.add(Dense(100,input_shape=(40,)))
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'))
```

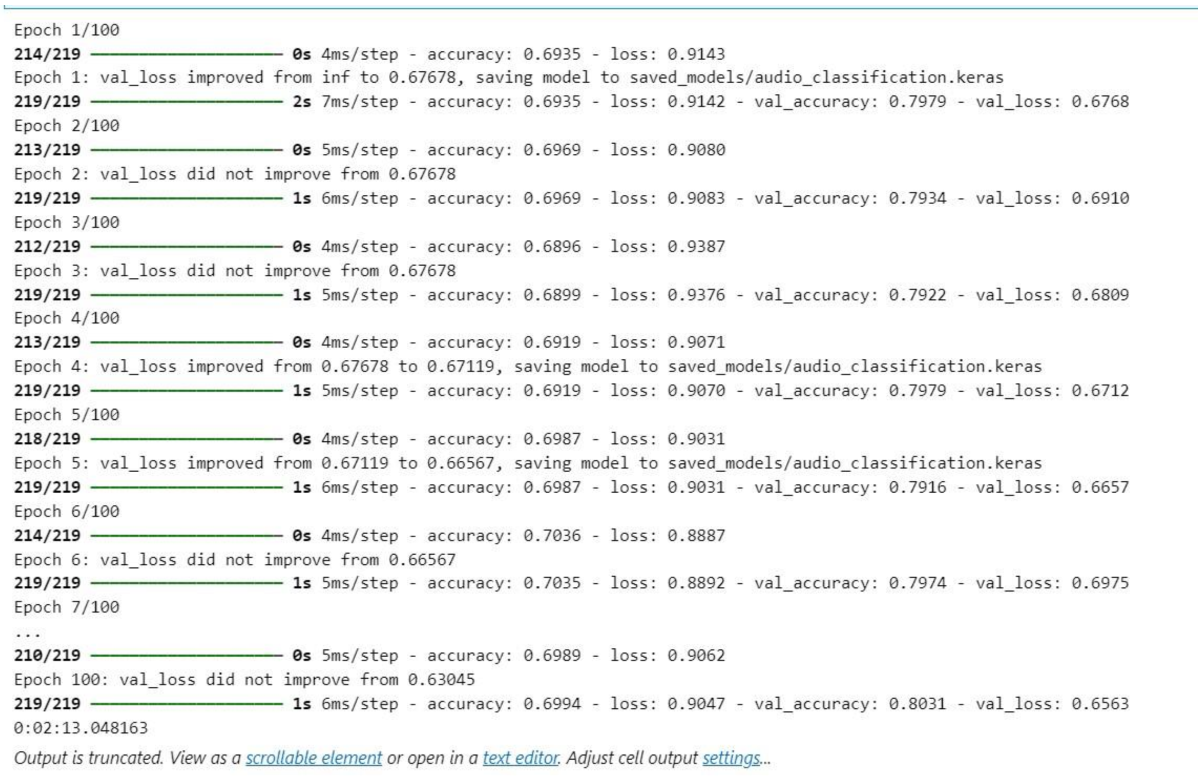


```
## training my model
from tensorflow.keras.callbacks import ModelCheckpoint

from datetime import datetime

num_epochs=100
num_batch_size=32

checkpointer=ModelCheckpoint(filepath='saved_models/audio_classification.keras',verbose=1,save_best_only=True)
start=datetime.now()
model.fit(x_train,y_train,batch_size=num_batch_size,epochs=num_epochs,validation_data=(x_test,y_test),callbacks=[checkpointer])
duration=datetime.now()-start
print(duration)
```

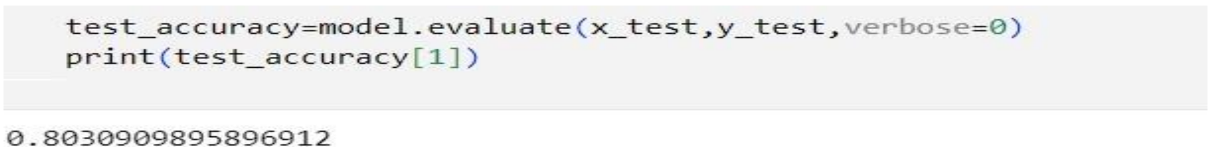


```
Epoch 1/100
214/219 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6935 - loss: 0.9143
Epoch 1: val_loss improved from inf to 0.67678, saving model to saved_models/audio_classification.keras
219/219 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6935 - loss: 0.9142 - val_accuracy: 0.7979 - val_loss: 0.6768
Epoch 2/100
213/219 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6969 - loss: 0.9080
Epoch 2: val_loss did not improve from 0.67678
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6969 - loss: 0.9083 - val_accuracy: 0.7934 - val_loss: 0.6910
Epoch 3/100
212/219 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6896 - loss: 0.9387
Epoch 3: val_loss did not improve from 0.67678
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6899 - loss: 0.9376 - val_accuracy: 0.7922 - val_loss: 0.6809
Epoch 4/100
213/219 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6919 - loss: 0.9071
Epoch 4: val_loss improved from 0.67678 to 0.67119, saving model to saved_models/audio_classification.keras
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6919 - loss: 0.9070 - val_accuracy: 0.7979 - val_loss: 0.6712
Epoch 5/100
218/219 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6987 - loss: 0.9031
Epoch 5: val_loss improved from 0.67119 to 0.66567, saving model to saved_models/audio_classification.keras
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6987 - loss: 0.9031 - val_accuracy: 0.7916 - val_loss: 0.6657
Epoch 6/100
214/219 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7036 - loss: 0.8887
Epoch 6: val_loss did not improve from 0.66567
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7035 - loss: 0.8892 - val_accuracy: 0.7974 - val_loss: 0.6975
Epoch 7/100
...
210/219 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6989 - loss: 0.9062
Epoch 100: val_loss did not improve from 0.63045
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6994 - loss: 0.9047 - val_accuracy: 0.8031 - val_loss: 0.6563
0:02:13.048163
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

•<u>**Neural network:**</u>

•<u>**Model accuracy:**</u>



```
test_accuracy=model.evaluate(x_test,y_test,verbose=0)
print(test_accuracy[1])
```

```
0.8030909895896912
```

Conclusion & Recommendations

Conclusion:

The project successfully identified that the voice in the call recordings and the lab-recorded voice belonged to the same individual. This conclusion was reached through rigorous forensic analysis using advanced voice recognition techniques, ensuring reliable results.

Recommendations:

1. **Regular Audits and Updates:**
   - Keep forensic tools updated to ensure functionality and security.
   - Conduct security audits to address vulnerabilities.
2. **Network Security:**
   - Use firewalls and VPNs to secure remote access and reduce cyber threats.
3. **Data Backup and Recovery:**
   - Implement regular backups and have a disaster recovery plan in place to mitigate data loss risks.
4. **Model Accuracy Improvements:**
   - Explore the use of convolutional neural networks (CNNs) to improve model accuracy.

Challenges and Limitations

- **Data Quality**:
  High-quality, diverse, and well-labeled data is essential for reliable models. Poor data quality leads to reduced performance.
- **Computational Resources:**
  Training complex models, especially deep neural networks, requires substantial resources, which can limit scalability.
- **Model Interpretability**:
  Neural networks often lack transparency, making it difficult to explain their decision-making process.
- **Scalability:**
  As tasks grow in complexity, maintaining performance and efficiency can become a significant challenge.

THANKYOU