Anchal 102103227 4CO15

QUES1 :- The paper presents the Speech Commands dataset, a collection of spoken words designed to aid in training and evaluating keyword spotting systems for voice interfaces. It addresses the challenges of on-device speech recognition, such as energy efficiency and the need to minimize false positives. The dataset, which includes a variety of accents and is released under a Creative Commons license, aims to standardize the training and evaluation process for keyword spotting models, making it accessible to a broader research and development community. The paper also discusses the collection process, the importance of background noise in training, and the release of baseline model results.

## Snapshots of Code

## Link for colab:-

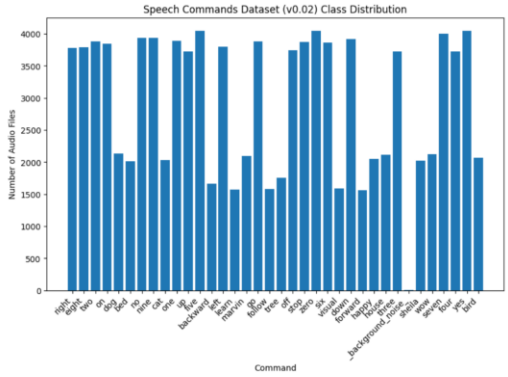
https://colab.research.google.com/drive/1LyH8hXLIsO3w30F5kdkcThyZAbxb70Nq?usp=sharing

**Link of my own dataset**

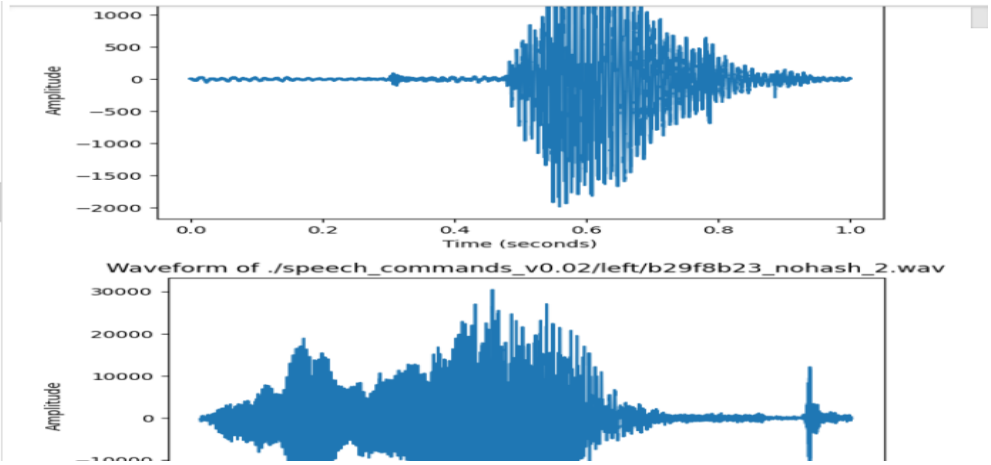
https://drive.google.com/drive/folders/1gqgu-_hDAwJ5VeO47StuOG-VHVWZvE9J

```
[9]  # Download URL (replace with the correct version if v0.02 is outdated)
     dataset_url = "http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz"

     # Download and extract the dataset
     data_dir = "./speech_commands_v0.02"
     if not os.path.exists(data_dir):
         os.makedirs(data_dir)
         print("Downloading dataset...")
         !wget {dataset_url} -P .
         print("Extracting dataset...")
         with tarfile.open(dataset_url.split("/")[-1], "r:gz") as tar:
             tar.extractall(data_dir)
             print("Dataset extraction complete!")
```

```
Downloading dataset...
--2024-09-11 05:57:29--  http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 74.125.195.207, 172.253.117.207, 142.250.99.207,
Connecting to download.tensorflow.org (download.tensorflow.org)|74.125.195.207|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2428923189 (2.3G) [application/gzip]
Saving to: ‘./speech_commands_v0.02.tar.gz’

speech_commands_v0. 100%[===================>]   2.26G  5.28MB/s    in 37s

2024-09-11 05:58:07 (62.5 MB/s) - ‘./speech_commands_v0.02.tar.gz’ saved [2428923189/2428923189]

Extracting dataset...
Dataset extraction complete!
```




```
Audio length statistics:
Minimum: 821 bytes
Maximum: 3045984 bytes
Average: 31552.46465285914 bytes
Standard deviation: 16264.196594505445 bytes
```




```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - accuracy: 0.0000e+00 - loss: 3.5494 - val_accuracy: 0.0000e+00 - val_loss: 81.9565
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 140.2354
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 183.8691
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 218.3487
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 245.7756
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 268.6526
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 288.7154
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 306.5362
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 322.4672
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 336.7675
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.0000e+00 - loss: 336.7675
Test accuracy: 0.00
```

```python
from sklearn.metrics import classification_report, confusion_matrix

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)

# Calculate classification report
report = classification_report(np.argmax(y_test, axis=1), y_pred_classes, target_names=label_encoder.classes_)
print("Classification Report:\n", report)

# Calculate confusion matrix
confusion_mat = confusion_matrix(np.argmax(y_test, axis=1), y_pred_classes)
print("Confusion Matrix:\n", confusion_mat)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 639ms/step
Classification Report:
                              precision    recall  f1-score   support

left_00176480_nohash_0.wav       0.00      0.00      0.00       0.0
left_004ae714_nohash_0.wav       0.00      0.00      0.00       1.0

                  accuracy                           0.00       1.0
                 macro avg       0.00      0.00      0.00       1.0
              weighted avg       0.00      0.00      0.00       1.0

Confusion Matrix:
 [[0 0]
 [1 0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-sco
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Recall and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-sco
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Recall and F-score
  _warn_prf(average, modifier, msg_start, len(result))
```

# Q5-

## MY voice

```python
[25] from google.colab import drive
     import os

     # Mount the Google Drive
     drive.mount('/content/drive')

     # Set the path to your dataset folder
     dataset_folder = 'https://drive.google.com/drive/folders/1gqgu-_hDAwJ5VeO47StuOG-VHVWZvE9J'  # Replace with the path to your dataset folder on Google Drive
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
from google.colab import drive
import os

# Mount the Google Drive
drive.mount('/content/drive')

# Set the path to your dataset folder
dataset_folder = '/content/drive/MyDrive/myVoice'  # Replace 'your_dataset_folder' with the actual folder name in your Drive

import librosa
import numpy as np

# Function to extract MFCC features from audio files
def extract_features(audio_path, n_mfcc=13):
    audio, sample_rate = librosa.load(audio_path, sr=None)
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)
    return np.mean(mfcc.T, axis=0)

# Initialize lists to hold data and labels
X = []
y = []

# Iterate over the files in your dataset folder
for command_dir in os.listdir(dataset_folder):
    command_path = os.path.join(dataset_folder, command_dir)
    if os.path.isdir(command_path):
        for filename in os.listdir(command_path):
            if filename.endswith('.wav'):  # Ensure you only process audio files
                audio_path = os.path.join(command_path, filename)
```

```
# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Check if the folder exists
import os
dataset_folder = '/content/drive/MyDrive/myVoice'

if not os.path.exists(dataset_folder):
    print("Dataset folder does not exist!")
else:
    print("Dataset folder found!")
```

```
Dataset folder found!
```

```
for command_dir in os.listdir(dataset_folder):
    command_path = os.path.join(dataset_folder, command_dir)
    if os.path.isdir(command_path):
        print(f"Checking folder: {command_dir}")
        files = os.listdir(command_path)
        print(f"Number of files in {command_dir}: {len(files)}")
        for filename in files:
            if filename.endswith('.wav'):
                print(f"Found audio file: {filename}")
```

```
Checking folder: myVoiceupadte
Number of files in myVoiceupadte: 29
Found audio file: down.wav
Found audio file: yes.wav
Found audio file: two.wav
Found audio file: three.wav
Found audio file: undo.wav
Found audio file: seven.wav
Found audio file: stop_fake.wav
Found audio file: six.wav
Found audio file: ten.wav
Found audio file: setting.wav
Found audio file: search.wav
Found audio file: right.wav
Found audio file: redo.wav
Found audio file: play.wav
Found audio file: restart.wav
Found audio file: no.wav
Found audio file: no_fake.wav
```

```python
import librosa

# Function to extract MFCC features from audio files
def extract_features(audio_path, n_mfcc=13):
    audio, sample_rate = librosa.load(audio_path, sr=None)
    print(f"Loaded audio file: {audio_path}, Sample Rate: {sample_rate}, Audio Shape: {audio.shape}")
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)
    print(f"MFCC shape: {mfcc.shape}")
    return np.mean(mfcc.T, axis=0)

# Test a specific audio file
test_audio_path = '/content/drive/MyDrive/myVoice/myVoiceupadte/five.wav'
extract_features(test_audio_path)
```

```
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/five.wav, Sample Rate: 48000, Audio Shape: (141120,)
MFCC shape: (13, 276)
array([-466.15436  ,   35.036575 ,  -12.235614 ,   10.087469 ,
        -4.5965962,   -2.5560079,   -7.0382614,   -1.3597052,
        -6.7001424,   -4.4848237,   -6.9268036,   -6.2571983,
        -3.9304097], dtype=float32)
```

```python
import librosa
import os
import numpy as np

# Function to extract MFCC features from audio files
def extract_features(audio_path, n_mfcc=13):
    audio, sample_rate = librosa.load(audio_path, sr=None)
    print(f"Loaded audio file: {audio_path}, Sample Rate: {sample_rate}, Audio Shape: {audio.shape}")
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)
    print(f"MFCC shape: {mfcc.shape}")
    return np.mean(mfcc.T, axis=0)

# Dataset folder path (change to your folder)
dataset_folder = '/content/drive/MyDrive/myVoice'  # Update to match your folder structure

# Initialize lists to store MFCC features and labels
X = []  # For features
y = []  # For labels (which could be folder names)

# Iterate over the files in your dataset folder
for command_dir in os.listdir(dataset_folder):
    command_path = os.path.join(dataset_folder, command_dir)
    if os.path.isdir(command_path):  # Ensure it's a directory
        print(f"Processing folder: {command_dir}")
        for filename in os.listdir(command_path):
            if filename.endswith('.wav'):  # Process only .wav files
                audio_path = os.path.join(command_path, filename)
                print(f"Processing file: {filename}")
                mfcc = extract_features(audio_path)  # Extract MFCC features
```

```
                print(f"Processing file: {filename}")
                mfcc = extract_features(audio_path)  # Extract MFCC features
                X.append(mfcc)  # Add the MFCC features to the list
                y.append(command_dir)  # Use the folder name as the label

# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)

# Check the shapes
print(f"Shape of X: {X.shape}, Shape of y: {y.shape}")
```

```
Processing folder: myVoiceupadte
Processing file: down.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/down.wav, Sample Rate: 48000, Audio Shape: (161280,)
MFCC shape: (13, 316)
Processing file: yes.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/yes.wav, Sample Rate: 48000, Audio Shape: (164160,)
MFCC shape: (13, 321)
Processing file: two.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/two.wav, Sample Rate: 48000, Audio Shape: (213120,)
MFCC shape: (13, 417)
Processing file: three.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/three.wav, Sample Rate: 48000, Audio Shape: (161280,)
MFCC shape: (13, 316)
Processing file: undo.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/undo.wav, Sample Rate: 48000, Audio Shape: (169920,)
MFCC shape: (13, 332)
Processing file: seven.wav
Loaded audio file: /content/drive/MyDrive/myVoice/myVoiceupadte/seven.wav, Sample Rate: 48000, Audio Shape: (164160,)
MFCC shape: (13, 321)
```

```
from sklearn.preprocessing import LabelEncoder

# Encode the labels into numerical format
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

print(f"Encoded labels: {y_encoded}")
print(f"Label classes: {label_encoder.classes_}")
```

```
Encoded labels: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Label classes: ['myVoiceupadte']
```

```
[39] from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

print(f"Shape of X_train: {X_train.shape}, Shape of X_test: {X_test.shape}")
```

```
Shape of X_train: (23, 13), Shape of X_test: (6, 13)
```

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define a simple neural network model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),  # Input layer
    Dense(64, activation='relu'),  # Hidden layer
    Dense(len(np.unique(y_encoded)), activation='softmax')  # Output layer (softmax for classification)
])

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

# Train the model
history = model.fit(X_train, y_train, epochs=30, batch_size=32, validation_data=(X_test, y_test))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/ops/nn.py:545: UserWarning: You are using a softmax over axis -1 of a tensor of shape (None, 1). This axis has
  warnings.warn(
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 0.0000e+00
Epoch 2/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 0.0000e+00
Epoch 3/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 0.0000e+00
Epoch 4/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_loss: 0.0000e+00
Epoch 5/30
```

```python
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc:.2f}")
```

```
1/1 - 0s - 33ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00
Test accuracy: 0.00
```

```python
import numpy as np

# Make predictions on the test set
y_pred = np.argmax(model.predict(X_test), axis=1)

# Print classification report
print("Classification Report:\n", classification_report(y_test, y_pred, target_names=label_encoder.classes_))

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
Classification Report:
                precision    recall  f1-score   support

myVoiceupadte       1.00      1.00      1.00         6

     accuracy                           1.00         6
    macro avg       1.00      1.00      1.00         6
 weighted avg       1.00      1.00      1.00         6

Confusion Matrix:
 [[6]]
/usr/local/lib/python3.10/dist-packages/keras/src/ops/nn.py:545: UserWarning: You are using a softmax over axis -1 of a ten
```