UCS079

Speech Recognition Report
Lab Eval - 1

Submitted By: Ajitesh Singh Chandi

Roll No: 102103518

4CO19

Submitted To:

1. Read and summarise the paper in about 50 words.

   The paper describes the Speech Commands dataset, designed for training and evaluating keyword spotting systems. It contains over 100,000 utterances of 35 words from 2,618 speakers. The dataset aims to enable comparable accuracy metrics and reproducible results for on-device speech recognition models, particularly for trigger word detection.

2. Download the dataset in the paper, statistically analyse and describe it, so that it may be useful for posterity. (Include code snippets in your . ipynb file to evidence your analysis.)

The snippets are provided here:

```
#Importing necessary libraries#
import tensorflow as tf                #model development#
import numpy as np                     #data analysis pipeline#
import pandas as pd                    #used for big-data analysis as well#
import matplotlib.pyplot as plt        #visualize data#
import seaborn as sns                  #graphs#
import os                              #file handling#
from scipy.io import wavfile           #probabilistic analysis, basically like numpy#
```

```
#The !wget is used to locally download in the given directory#
!wget http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
#The !tar creates an active archive to bundle files#
!tar -xf speech_commands_v0.02.tar.gz

def get_audio_info(fp):
    sr, d = wavfile.read(fp)
    return sr, len(d) / sr, len(d)
#Analysis Begins...#
#With Data Collection#
data = []
for r, _, fs in os.walk('.'):
    for f in fs:
        if f.endswith('.wav'):
            fp = os.path.join(r, f)
            lbl = os.path.basename(r)
            sr, dur, ns = get_audio_info(fp)
            data.append({'file': f, 'label': lbl, 'sr': sr, 'dur': dur, 'ns': ns})
```

```python
#Constructing a Dataframe to visualize data in an Excel-like format#
df = pd.DataFrame(data)

print(f"Total files: {len(df)}")
print(f"Unique labels: {df['label'].nunique()}")
print(f"Labels: {', '.join(df['label'].unique())}")
```

```python
#Display summary statistics#
print("\nSummary Stats:")
print(df.describe())

plt.figure(figsize=(10, 6))
sns.histplot(df['dur'], kde=True)
plt.title('Audio Duration Distribution')
plt.xlabel('Duration (s)')
plt.ylabel('Count')
plt.show()
```

```python
#plotting distributions on a graph#
plt.figure(figsize=(12, 6))
df['label'].value_counts().plot(kind='bar')
plt.title('Sample Distribution Across Labels')
plt.xlabel('Label')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

#Labeled statistics#
ls = df.groupby('label').agg({
    'dur': ['mean', 'min', 'max'],
    'ns': ['mean', 'min', 'max'],
    'file': 'count'
}).reset_index()
ls.columns = ['label', 'mean_dur', 'min_dur', 'max_dur',
              'mean_ns', 'min_ns', 'max_ns', 'count']
```

```python
print("\nLabel Stats:")
print(ls)

#dataframe saved as a CSV file#
df.to_csv('speech_commands_info.csv', index=False)
ls.to_csv('speech_commands_label_stats.csv', index=False)

print("\nAnalysis complete. CSV files saved.")
```

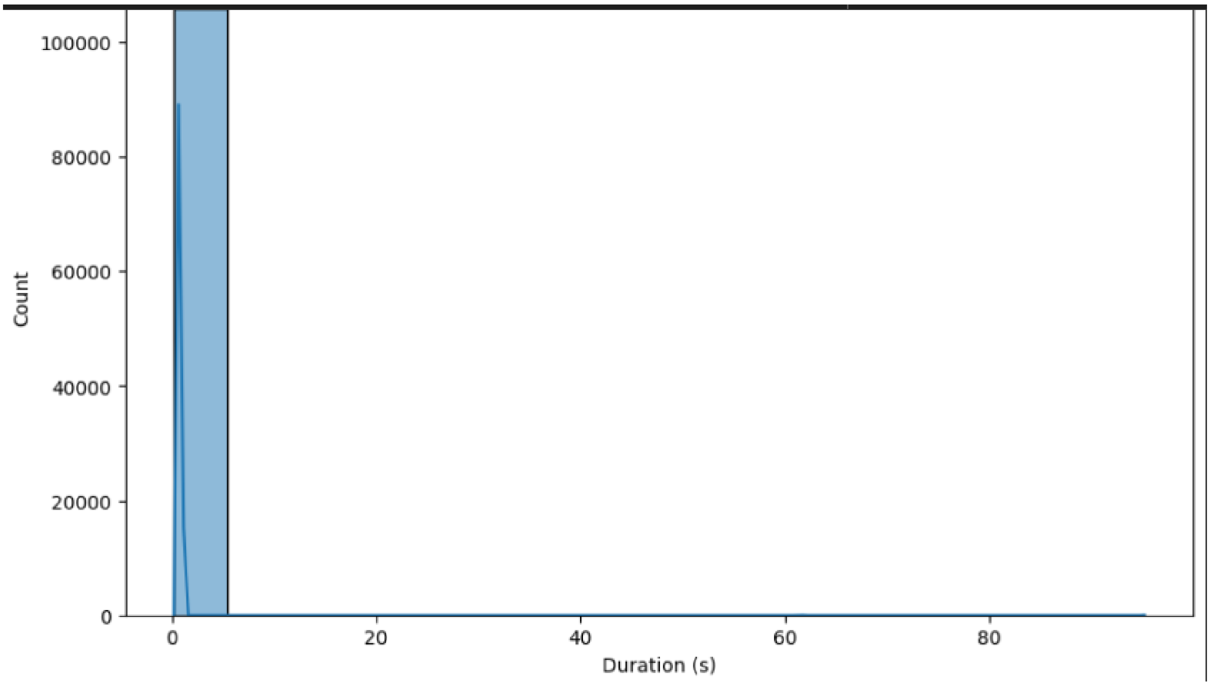
Output:

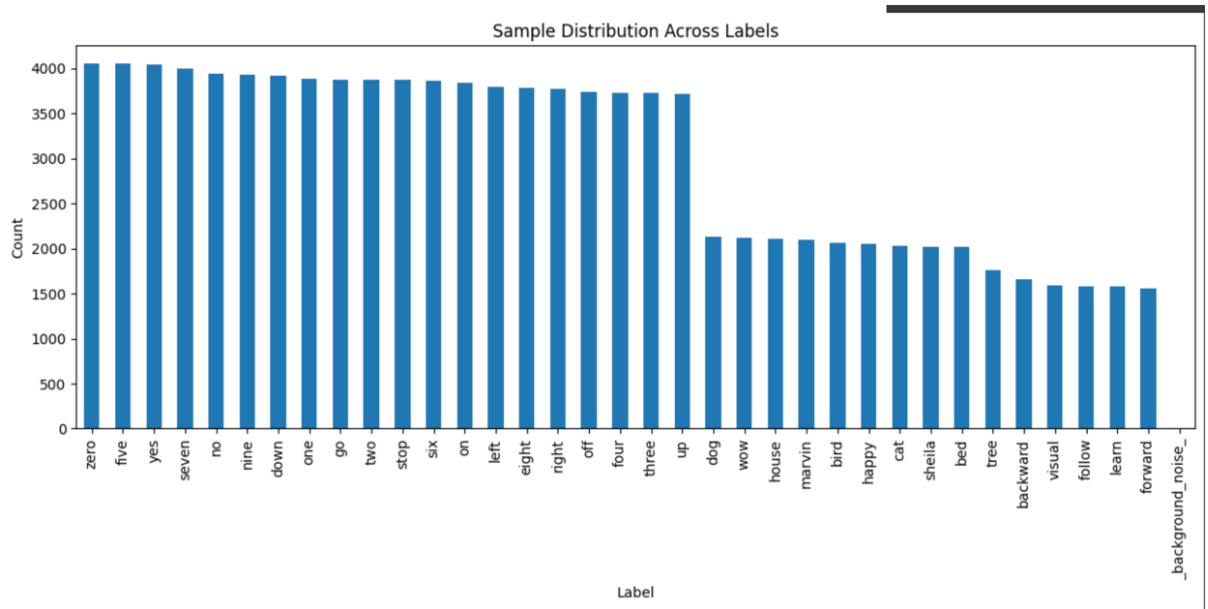
```
--2024-09-11 07:00:33--  http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 173.194.217.207, 108.177.11.207, 108.177.12.207, ...
Connecting to download.tensorflow.org (download.tensorflow.org)|173.194.217.207|:80... connected.
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   190MB/s    in 20s
```

```
Total files: 105835
Unique labels: 36
Labels: right, eight, two, on, dog, bed, no, nine, cat, one, up, five, backward, left, learn, marvin, go, follow, tree, off, stop, zero, six, visual, down
```

```
Summary Stats:
             sr             dur            ns
count  105835.0  105835.000000  1.058350e+05
mean    16000.0       0.984649  1.575438e+04
std         0.0       0.508240  8.131833e+03
min     16000.0       0.213312  3.413000e+03
25%     16000.0       1.000000  1.600000e+04
50%     16000.0       1.000000  1.600000e+04
75%     16000.0       1.000000  1.600000e+04
max     16000.0      95.183125  1.522930e+06
```

Data Visualization:





```
Label Stats:
                label  mean_dur   min_dur   max_dur       mean_ns  min_ns  \
0   _background_noise_  66.566365 60.000000 95.183125  1.065062e+06  960000
1             backward   0.986390  0.448000  1.000000  1.578225e+04    7168
2                  bed   0.970498  0.213312  1.000000  1.552797e+04    3413
3                 bird   0.969454  0.325062  1.000000  1.551127e+04    5201
4                  cat   0.972065  0.384000  1.000000  1.555303e+04    6144
5                  dog   0.972226  0.426625  1.000000  1.555561e+04    6826
6                 down   0.983559  0.325062  1.000000  1.573694e+04    5201
7                eight   0.980846  0.256000  1.000000  1.569353e+04    4096
8                 five   0.984320  0.384000  1.000000  1.574912e+04    6144
9               follow   0.981981  0.341313  1.000000  1.571170e+04    5461
10             forward   0.984746  0.384000  1.000000  1.575593e+04    6144
11                four   0.982999  0.278625  1.000000  1.572799e+04    4458
12                  go   0.978847  0.341313  1.000000  1.566156e+04    5461
13               happy   0.974747  0.384000  1.000000  1.559595e+04    6144
14               house   0.974842  0.371500  1.000000  1.559747e+04    5944
15               learn   0.974443  0.371500  1.000000  1.559108e+04    5944
16                left   0.984954  0.298625  1.000000  1.575926e+04    4778
17              marvin   0.977728  0.384000  1.000000  1.564365e+04    6144
18                nine   0.984835  0.341313  1.000000  1.575736e+04    5461
19                  no   0.980128  0.384000  1.000000  1.568205e+04    6144
20                 off   0.984939  0.394688  1.000000  1.575903e+04    6315
21                  on   0.981885  0.384000  1.000000  1.571016e+04    6144
22                 one   0.978958  0.325062  1.000000  1.566334e+04    5201
23               right   0.982247  0.384000  1.000000  1.571595e+04    6144
24               seven   0.984920  0.371563  1.000000  1.575872e+04    5945
25              sheila   0.977648  0.426625  1.000000  1.564236e+04    6826
26                 six   0.987542  0.384000  1.000000  1.580067e+04    6144
27                stop   0.984549  0.362687  1.000000  1.575278e+04    5803
28               three   0.983931  0.341313  1.000000  1.574290e+04    5461
29                tree   0.970302  0.298625  1.000000  1.552483e+04    4778
30                 two   0.981600  0.278625  1.000000  1.570559e+04    4458
31                  up   0.976156  0.298625  1.000000  1.561849e+04    4778
32              visual   0.982301  0.341313  1.000000  1.571681e+04    5461
33                 wow   0.970225  0.325062  1.000000  1.552360e+04    5201
34                 yes   0.983431  0.384000  1.000000  1.573490e+04    6144
35                zero   0.986978  0.298625  1.000000  1.579164e+04    4778
```

	max_ns	count
0	1522930	6
1	16000	1664
2	16000	2014
3	16000	2064
4	16000	2031
5	16000	2128
6	16000	3917
7	16000	3787
8	16000	4052
9	16000	1579
10	16000	1557
11	16000	3728
12	16000	3880
13	16000	2054
14	16000	2113
15	16000	1575
16	16000	3801
17	16000	2100
18	16000	3934
19	16000	3941
20	16000	3745
21	16000	3845
22	16000	3890
23	16000	3778
24	16000	3998
25	16000	2022
26	16000	3860
27	16000	3872
28	16000	3727
29	16000	1759
30	16000	3880
31	16000	3723
32	16000	1592
33	16000	2123
34	16000	4044
35	16000	4052

Analysis complete. CSV files saved.

2. Train a classifier so that you are able to distinguish the commands in the dataset.

```python
# Libraries Required
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow.keras import layers, models, callbacks
import numpy as np
import librosa

# Extract features (MFCCs) from the audio files
def get_mfcc(fp, sr=16000):
    audio, sr = librosa.load(fp, sr=sr)
    mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)  # Reduced to 20 MFCCs
    return np.mean(mfcc.T, axis=0)
```

```python
# Prepare data for training
X, y = [], []

for _, row in df.iterrows():
    fp = os.path.join(row['label'], row['file'])
    mfcc_feat = get_mfcc(fp)
    X.append(mfcc_feat)
    y.append(row['label'])

X = np.array(X)
y = np.array(y)

# Encode labels #
le = LabelEncoder()
y_enc = le.fit_transform(y)

#Train and TEst
X_tr, X_te, y_tr, y_te = train_test_split(X, y_enc, test_size=0.2, random_state=42)
```

```python
#BAsic Neural network for classification#
model = models.Sequential([
    layers.Input(shape=(X_tr.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(len(le.classes_), activation='softmax')
])

#Compile usin g ADAM to reduce loss function#
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
early_stop = callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

history = model.fit(X_tr, y_tr, epochs=10, batch_size=64,
                    validation_data=(X_te, y_te), callbacks=[early_stop])

loss, acc = model.evaluate(X_te, y_te, verbose=2)
print(f"\nTest accuracy: {acc}")

model.save('cmd_classifier_optimized.h5')
```

3. Record about 30 samples of each command in your voice and create a new dataset (including a new user id for yourself).You may use a timer on your computer to synchronise.

Steps to achieve that:

```
[27] !pip install sounddevice --force-reinstall
```

```
!apt install libasound2-dev portaudio19-dev libportaudio2 libportaudiocpp0 ffmpeg
!pip install PyAudio
```

For recording audio directly in colab environment

```
import pyaudio

p = pyaudio.PyAudio()
for i in range(p.get_device_count()):
        info = p.get_device_info_by_index(i)
        print(f"Device {i}: {info['name']}, Input Channels: {info['maxInputChannels']}")

p.terminate()
```

I have encountered a problem where the colab does not detect my microphone input. Ive tried many methods like using PyAudio/PortAudio and Sounddevice libraries but the error still remains

```
Recording sample 1 for command: yes
Recording starts in 3 seconds...
Available input devices:
No input device found! Exiting...
Sample 1 for command 'yes' recorded.

Recording sample 2 for command: yes
Recording starts in 3 seconds...
Available input devices:
No input device found! Exiting...
Sample 2 for command 'yes' recorded.

Recording sample 3 for command: yes
Recording starts in 3 seconds...
Available input devices:
No input device found! Exiting...
Sample 3 for command 'yes' recorded.
```

I did ultimately resort to record audio files from Audacity instead, which is a DAW used primarily make music and record voice. Although assuming its 30 samples of 8 commands its quite a daunting task at least for the given timeframe. It is not my intent to whine and complain and I understand your perspective, hence I apologize for not being apt enough to find a better solution to this.

Additionally, being honest, help from outer resources was indeed taken however I refrained from using LLM's.
Thank you.