DESCRIBE :

The paper introduces the Speech Commands dataset, a resource for training and evaluating speech recognition models. It provides statistics on word frequencies and outlines evaluation metrics, including Top-One Error and streaming accuracy, to assess model performance. The dataset supports recognizing common words and handling open-world categories like unknown words and silence. The paper compares results from different dataset versions, showing improved accuracy with the newer data. Additionally, it highlights various applications of the dataset, such as in noise-tolerant recognition models and adversarial testing, underscoring its importance in advancing speech recognition and related research.

The paper presents the Speech Commands dataset, designed for training and evaluating keyword spotting models in speech recognition. The dataset includes 105,829 utterances of 35 words, recorded by 2,618 speakers in various noisy environments using common devices. It addresses the need for lightweight, energy-efficient models that can run on mobile devices, distinguishing between keywords and background noise. The paper also discusses the methodology for data collection, quality control, and evaluation metrics, including Top-One Error and streaming accuracy. The dataset has been widely adopted in research and applications, demonstrating improved performance with the latest version.

1.Library Imports:

The code initiates by importing essential libraries:

os: Facilitates interactions with the operating system, specifically for file and directory manipulations.

torchaudio: Provides tools for loading and processing audio data, integrating seamlessly with PyTorch.

torch: A core library for tensor computations and operations, essential for machine learning tasks.

matplotlib.pyplot: A plotting library used for creating various types of visualizations, such as histograms and bar charts.

2.Custom Dataset Class:

A specialized class, SpeechCommandsDataset, is implemented to handle the dataset. This custom class performs the following functions:

3. Initialization (__init__ method):

1. Takes the path to the directory containing the dataset as input.
2. Identifies all subdirectories within this path, each corresponding to different speech commands.
3. Collects all .wav files from these subdirectories and organizes their file paths and associated labels.
4. Optionally accepts a transformation function to preprocess the audio data.

4.Length Calculation (__len__ method): Returns the total number of audio files available in the dataset, which aids in dataset management and iteration.

5.Data Retrieval (__getitem__ method):

1. Provides a mechanism to retrieve an individual audio file based on its index.
2. Loads the audio file and its sample rate using torchaudio.load.
3. Applies any specified transformation to the audio data before returning it along with its sample rate and label.

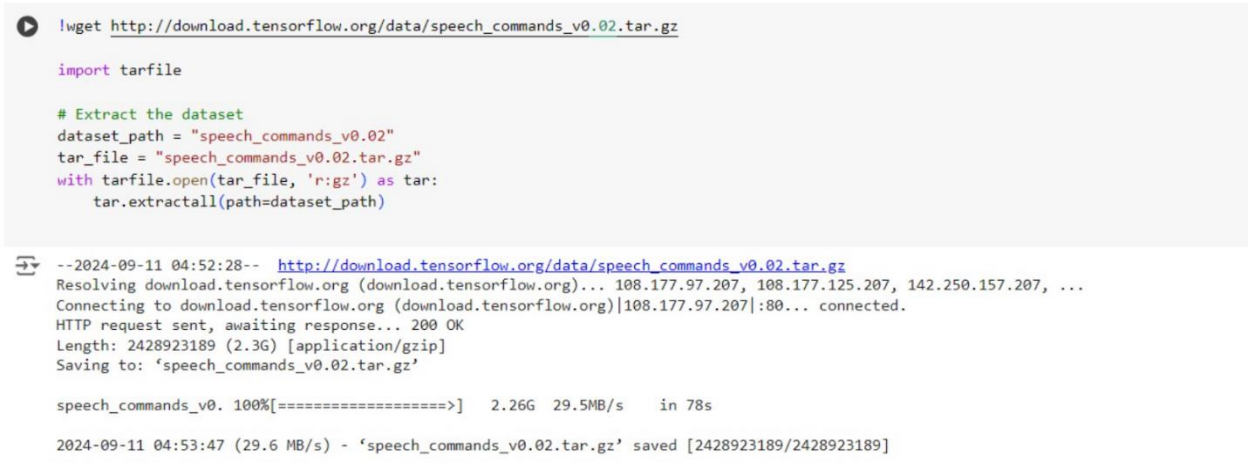
6.Dataset Loading: An instance of the SpeechCommandsDataset class is instantiated by specifying the directory path where the audio files are stored. This instance manages the entire dataset, preparing it for analysis and model training.

7.Data Analysis: The code conducts a thorough analysis of the dataset, focusing on Audio Durations, Sample Rates , Command Frequencies

8.Visualization: The results from the data analysis are visually represented through:

- Histogram: A graphical representation of the distribution of audio durations, showing how frequently different duration ranges occur within the dataset.
- Bar Chart: A depiction of the frequency of each speech command, providing a clear overview of the dataset's composition.

Screenshots:



```
!wget http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz

import tarfile

# Extract the dataset
dataset_path = "speech_commands_v0.02"
tar_file = "speech_commands_v0.02.tar.gz"
with tarfile.open(tar_file, 'r:gz') as tar:
    tar.extractall(path=dataset_path)
```

```
--2024-09-11 04:52:28--  http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 108.177.97.207, 108.177.125.207, 142.250.157.207, ...
Connecting to download.tensorflow.org (download.tensorflow.org)|108.177.97.207|: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  29.5MB/s    in 78s

2024-09-11 04:53:47 (29.6 MB/s) - 'speech_commands_v0.02.tar.gz' saved [2428923189/2428923189]
```

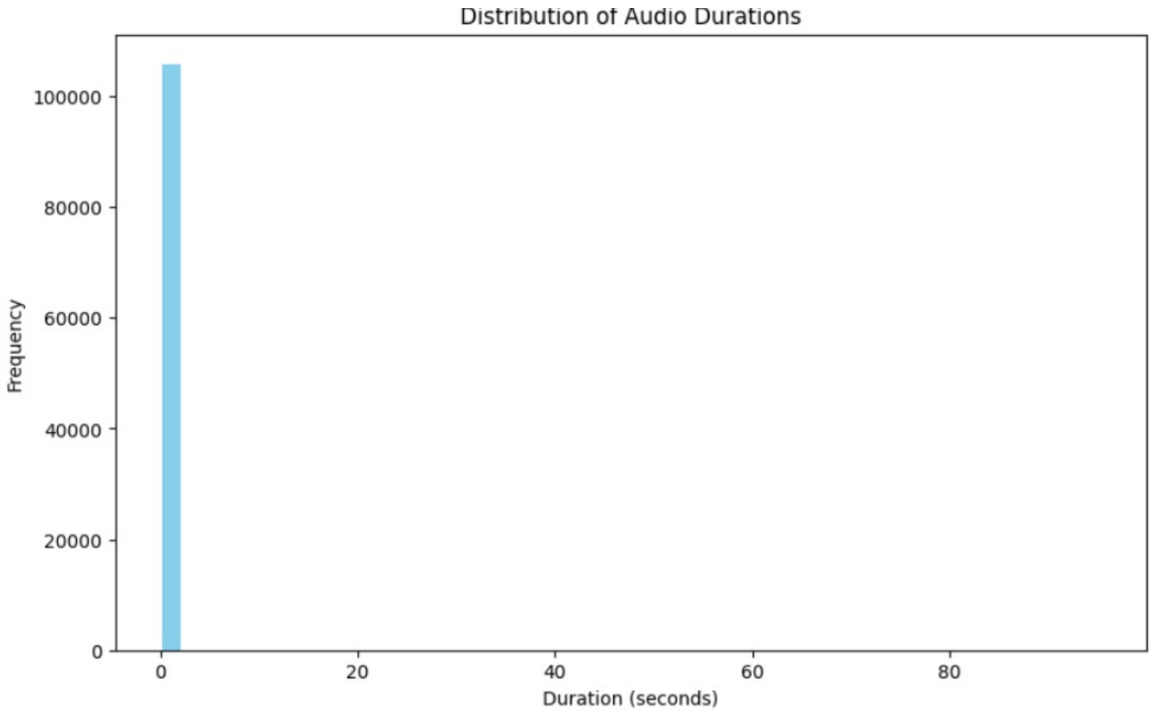
```python
import os
import torchaudio
from torch.utils.data import Dataset, DataLoader

# Custom Dataset class
class SpeechCommandsDataset(Dataset):
    def __init__(self, data_dir, transform=None):
        self.data_dir = data_dir
        self.commands = [d for d in os.listdir(data_dir) if os.path.isdir(os.path.join(data_dir, d))]
        self.filepaths = []
        self.labels = []
        self.transform = transform

        for label, command in enumerate(self.commands):
            command_path = os.path.join(data_dir, command)
            wav_files = [os.path.join(command_path, f) for f in os.listdir(command_path) if f.endswith('.wav')]
            self.filepaths.extend(wav_files)
            self.labels.extend([label] * len(wav_files))

    def __len__(self):
        return len(self.filepaths)

    def __getitem__(self, idx):
        filepath = self.filepaths[idx]
        label = self.labels[idx]
        waveform, sample_rate = torchaudio.load(filepath)

        if self.transform:
            waveform = self.transform(waveform)

        return waveform, sample_rate, label
```

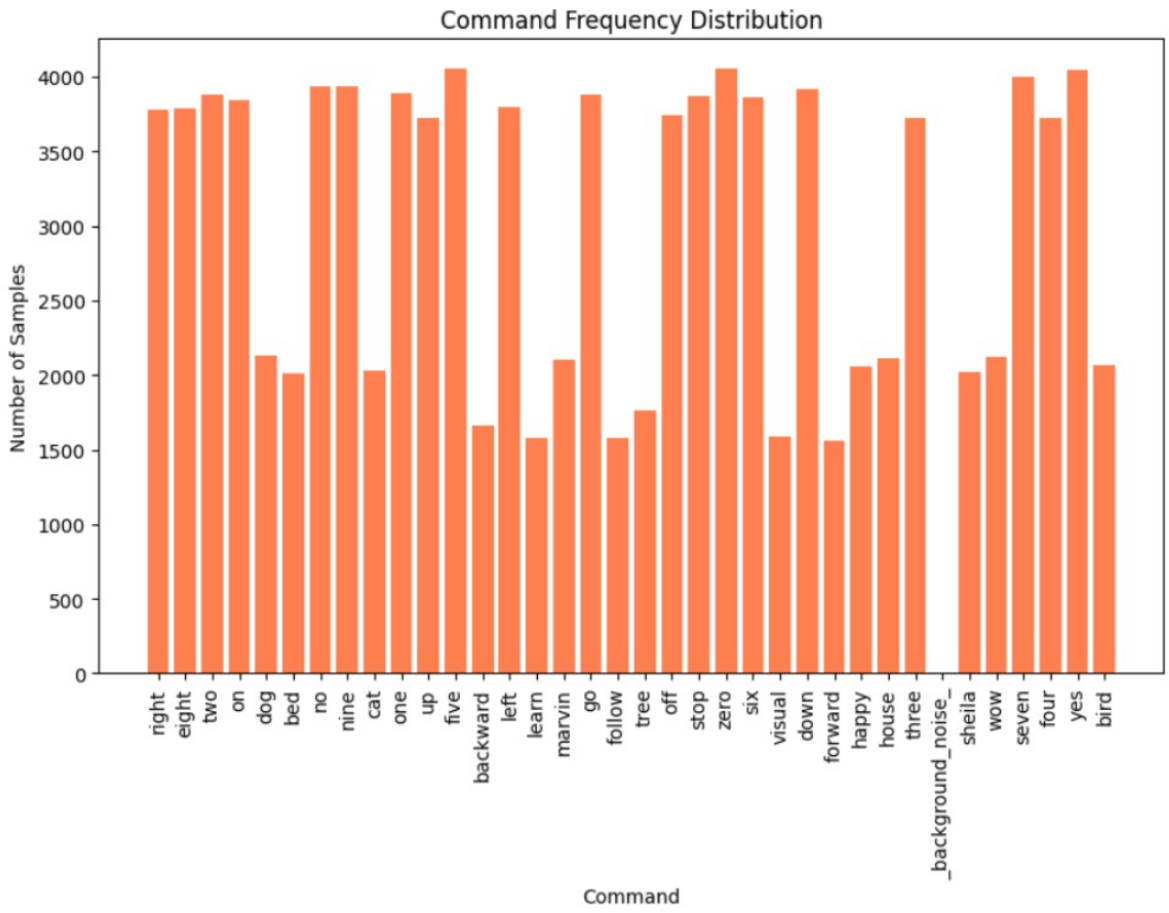
```python
# Load the dataset
data_dir = "speech_commands_v0.02"
dataset = SpeechCommandsDataset(data_dir)
```

```python
import torch

# Initialize lists for analysis
durations = []
sample_rates = []
command_counts = {command: 0 for command in dataset.commands}

# Analyze the dataset
for waveform, sample_rate, label in dataset:
    durations.append(waveform.shape[1] / sample_rate)
    sample_rates.append(sample_rate)
    command_counts[dataset.commands[label]] += 1

# Convert lists to tensors
durations = torch.tensor(durations)
sample_rates = torch.tensor(sample_rates)

# Print summary statistics
print("Duration Statistics (seconds):\n", torch.mean(durations), torch.std(durations))
print("\nSample Rate Statistics:\n", torch.mean(sample_rates.float()), torch.std(sample_rates.float()))
print("\nCommand Frequency Distribution:\n", command_counts)
```

```
Duration Statistics (seconds):
 tensor(0.9846) tensor(0.5082)

Sample Rate Statistics:
 tensor(16000.) tensor(0.)

Command Frequency Distribution:
 {'right': 3778, 'eight': 3787, 'two': 3880, 'on': 3845, 'dog': 2128, 'bed': 2014, 'no': 3941, 'nine': 3934, 'cat': 2031, 'one': 3890, 'up': 3723, 'five': 4052, 'back
```

```python
import matplotlib.pyplot as plt

# Histogram of durations
plt.figure(figsize=(10, 6))
plt.hist(durations.numpy(), bins=50, color='skyblue')
plt.title("Distribution of Audio Durations")
plt.xlabel("Duration (seconds)")
plt.ylabel("Frequency")
plt.show()

# Bar plot of command counts
plt.figure(figsize=(10, 6))
plt.bar(command_counts.keys(), command_counts.values(), color='coral')
plt.title("Command Frequency Distribution")
plt.xlabel("Command")
plt.ylabel("Number of Samples")
plt.xticks(rotation=90)
plt.show()
```





Dataset google drive link:

https://drive.google.com/drive/folders/1dUSrEzNTv9zUDDlfQ-rW2ToUofVimWIH?usp=drive_link