Conversational AI: Speech Recognition (UCS749)

Lab Evaluation I

Submitted By:

Akshat Garg **102116084** **4CS11**

Submitted To:

Dr. Raghav B. Venkataramaiyer

September, 2024

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

I. TASK DESCRIPTION

The objective of this lab evaluation is to work with the Speech Commands dataset, as detailed in the paper *An Overview of the Speech Commands Dataset*. The tasks include summarizing the paper, performing statistical analysis on the dataset, training a classifier, and fine-tuning it with custom-recorded samples.

Ia. TASK

Consider the paper *https://arxiv.org/abs/1804.03209*

- Read and summarize the paper in about 50 words.
- 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.)
- Train a classifier so that you are able to distinguish the commands in the dataset.
- Report the performance results using standard benchmarks.
- 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.
- Fine tune your classifier to perform on your voice.
- Report the results.

II. PAPER SUMMARY

The paper "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition" by Pete Warden describes a dataset designed for training and evaluating keyword spotting systems. It outlines the dataset's creation, containing spoken words to detect specific commands, and discusses its motivation, collection process, and evaluation methodology. The dataset, released under Creative Commons BY 4.0, includes various words, background noises, and is aimed at enabling accurate, energy-efficient, on-device speech recognition models.

III. DATA ANALYSIS

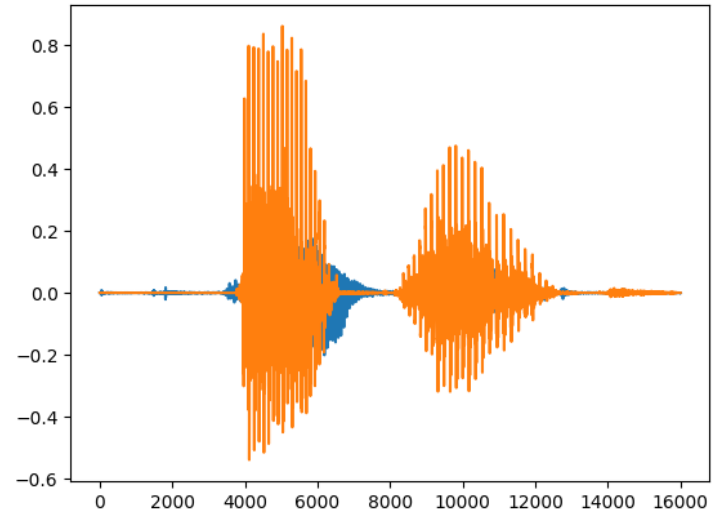
Downloading the Dataset

The Torchaudio library includes a built-in dataset for speech commands, containing over 84,000 samples of commands categorized into 35 labels.

```
from torchaudio.datasets import SPEECHCOMMANDS
```

Visualizing the Data and Samples

All the audio samples are 1 second long, with a sampling rate of 16,000Hz.



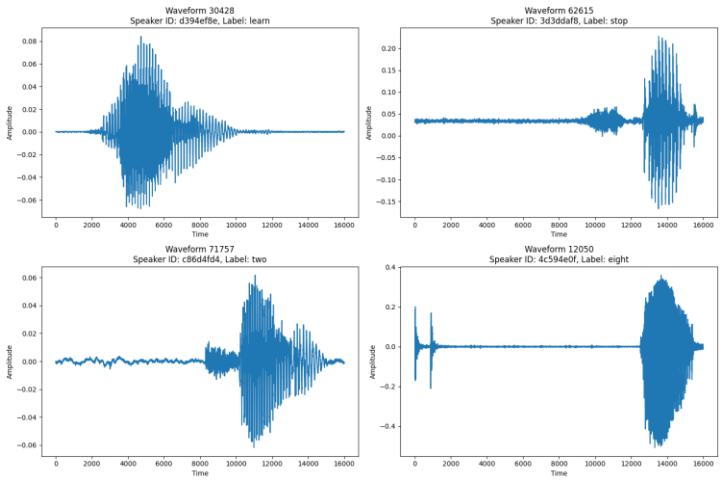


Figure: The waveforms of random samples

Statistical Analysis and Description

I processed the Speech-Commands dataset using Torchaudio, calculating and visualizing statistics such as mean, standard deviation, minimum, and maximum values for each sample, and analyzes these statistics by speaker.

Figure: Various Statistics Graphs

IV. TRAINING THE CLASSIFIER

Defining the Model

M5(
  (conv1): Conv1d(1, 32, kernel_size=(80,), stride=(16,))
  (bn1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(32, 32, kernel_size=(3,), stride=(1,))
  (bn2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool2): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv1d(32, 64, kernel_size=(3,), stride=(1,))
  (bn3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool3): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv4): Conv1d(64, 64, kernel_size=(3,), stride=(1,))
  (bn4): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool4): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=64, out_features=35, bias=True)
)
Number of parameters: 26915

optimizer = optim.Adam(model.parameters(), lr=0.01, weight_decay=0.0001)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.1)

Figure: Loss vs Batches Epochs*

Model Training and Performance Results

Test Epoch: 1 Accuracy: 6325/11005 (57%)
Test Epoch: 2 Accuracy: 6766/11005 (61%)
Test Epoch: 3 Accuracy: 7251/11005 (66%)
Test Epoch: 4 Accuracy: 8660/11005 (79%)
Test Epoch: 5 Accuracy: 8412/11005 (76%)
Test Epoch: 6 Accuracy: 8525/11005 (77%)
Test Epoch: 7 Accuracy: 8393/11005 (76%)
Test Epoch: 8 Accuracy: 8227/11005 (75%)
Test Epoch: 9 Accuracy: 8337/11005 (76%)
Test Epoch: 10 Accuracy: 8895/11005 (81%)

```python
import sounddevice as sd
import numpy as np
import scipy.io.wavfile as wav
import os

def record_audio(duration, sample_rate=8000):
    print("Recording...")
    audio = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=1, dtype='int16')
    sd.wait()  # Wait for the recording to finish
    return audio

words = ['backward','bed','bird','cat','dog','down','eight','five','follow','forward',
 'four','go','happy','house','learn','left','marvin','nine','no','off','on','one','right','seven',
 'sheila','six','stop','three','tree','two','up','visual','wow','yes','zero']
# Record a sample
for j in words:
    print(j)
    for kk in range(100):
        continue
    os.mkdir(j)
    for i in range(30):
        audio_sample = record_audio(1)
        wav.write(f"{j}/{i}.wav", 8000, audio_sample)
```

Figure: Recording Script

V. CUSTOM DATASET

Recording Custom Samples

The script used to record the speech commands includes instructions for pronouncing the words to be recorded in a comfortable manner. Consequently, all samples could have been collected within approximately 30 to 35 minutes.

Fine-tuning the Classifier

I utilized transfer learning and trained the model over 3 epochs, employing an 80-20 split of the 1,050 self-recorded samples.

Results

Test Epoch: 1 Accuracy: 176/210 (84%)
Test Epoch: 2 Accuracy: 197/210 (94%)
Test Epoch: 3 Accuracy: 210/210 (100%)

Demo Script

There is a demo script mentioned in the end for real time testing.

Submitted by:

Akshat Garg
102116084
4CS11