Lab Evaluation 1 for

UCS749 : Speech Processing and Synthesis

SUBMITTED BY

Aiman Gupta

102103488

PROJECT TITLE

<u>Recognise My Voice Commands</u>

COURSE COORDINATOR

Raghav B. Venkataramaiyer

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA

SESSION: July-December, 2024

DATE OF SUBMISSION: 21/03/2024

Summary of the paper

The paper introduces the "Speech Commands Dataset," a collection of spoken words for training keyword spotting systems. Since it is intended for low-resource devices, effective models must be able to understand commands such as "Yes," "No," and "Stop." The dataset, which has baseline models with an accuracy of 88.2%, aids in improving model comparability and reproducibility. More than 100,000 words from 2,618 speakers make up this collection.

Dataset

The Speech Commands Dataset described in the research paper contains over 105,829 audio recordings of 35 different words. The recordings are in WAV format, sampled at 16 kHz, and each audio file contains a single spoken word. The dataset was collected from 2,618 speakers, ensuring a wide variety of pronunciations and accents. It primarily focuses on small-vocabulary keyword spotting tasks, with words like "Yes," "No," "Up," "Down," "Left," "Right," "On," "Off," "Stop," "Go," and additional words such as digits (zero to nine).

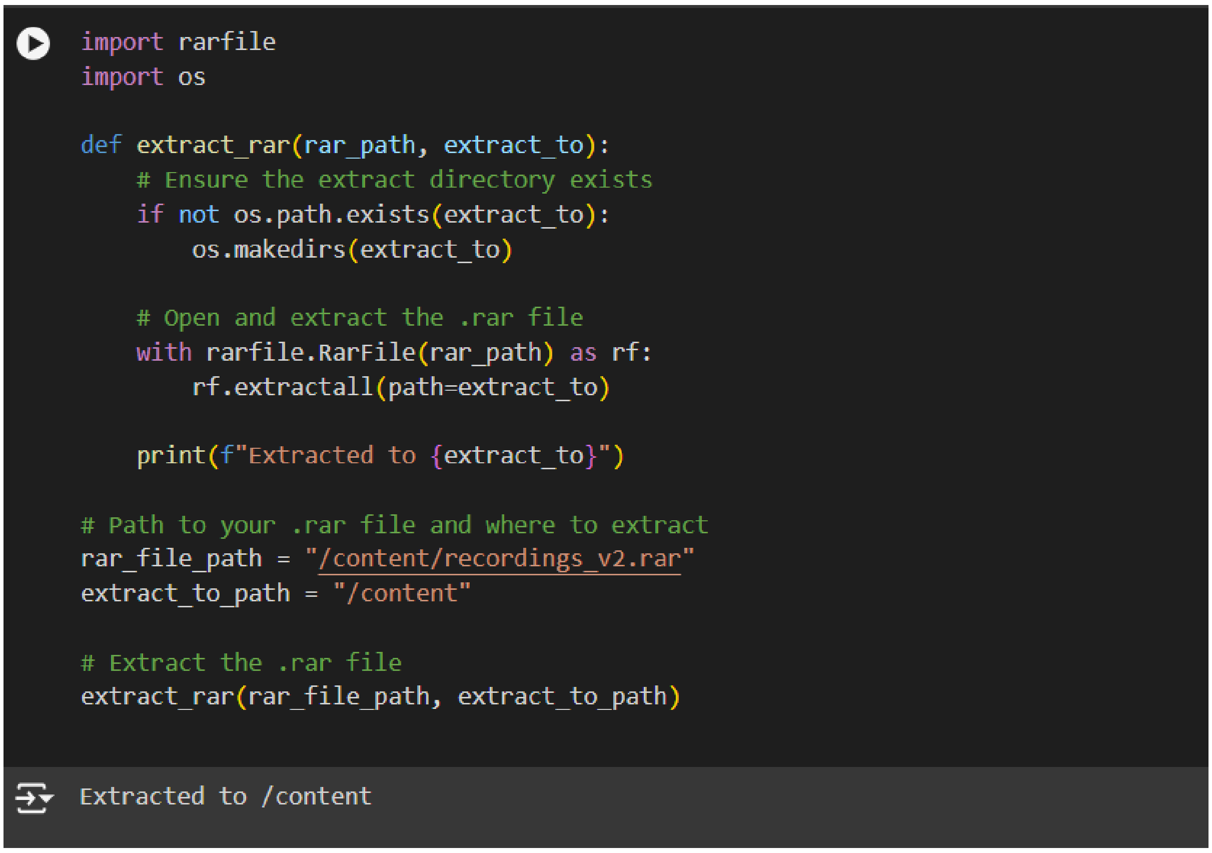
The entire dataset is approximately 3.8 GB when uncompressed, or 2.7 GB when stored as a gzip-compressed tar archive. Additionally, it includes background noise files to simulate real-world conditions and improve model robustness.

Key details:

- Words: 35 target words, including digits, commands, and auxiliary words.
- Recordings: 105,829 utterances.
- Speakers: 2,618 speakers, each assigned a unique identifier.
- File Format: 1-second WAV files, 16-bit PCM, 16 kHz sample rate.
- Size: ~3.8 GB uncompressed, ~2.7 GB compressed.

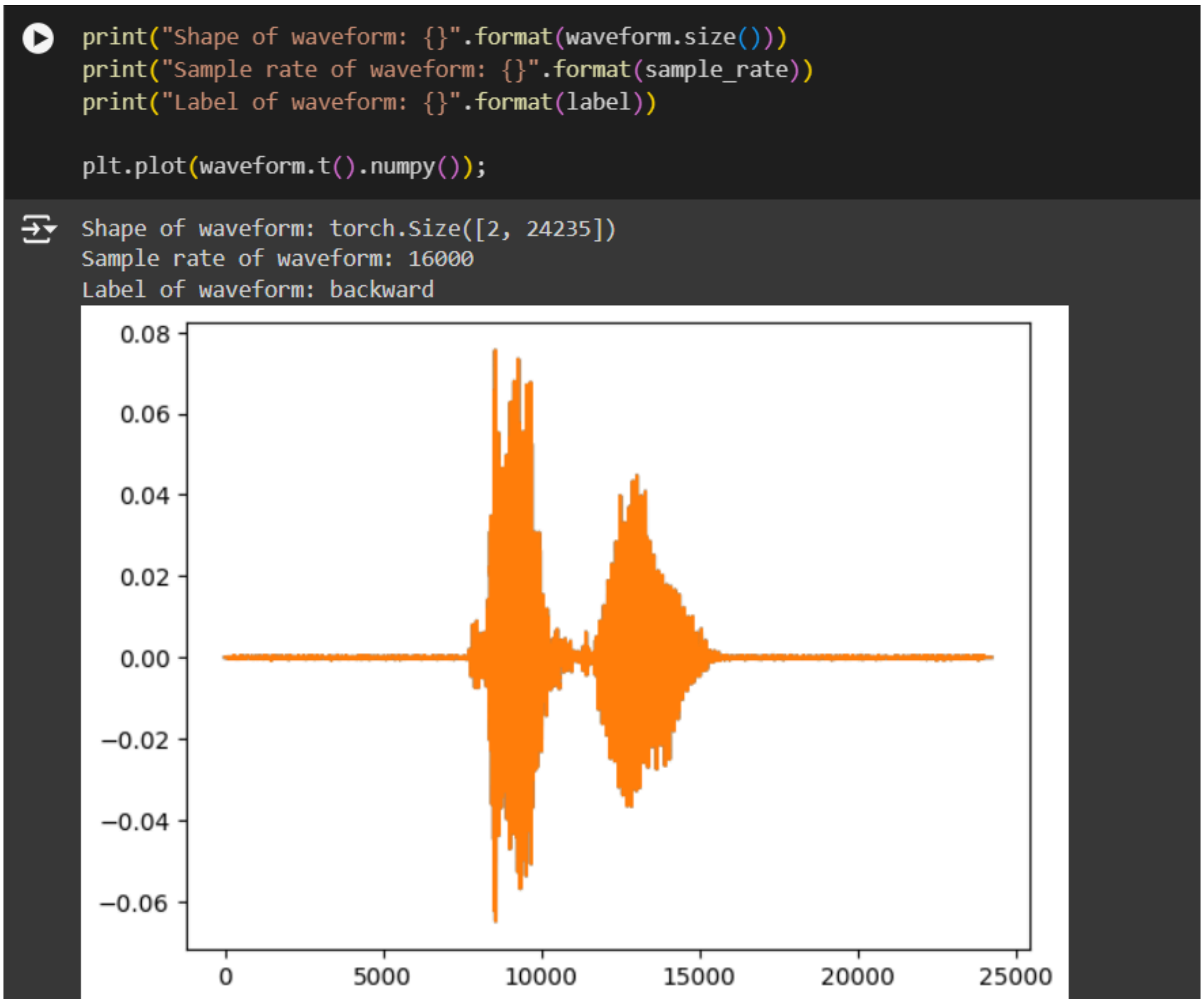
This dataset is designed to train and evaluate models for on-device keyword recognition in low-resource environments.

Snapshots :



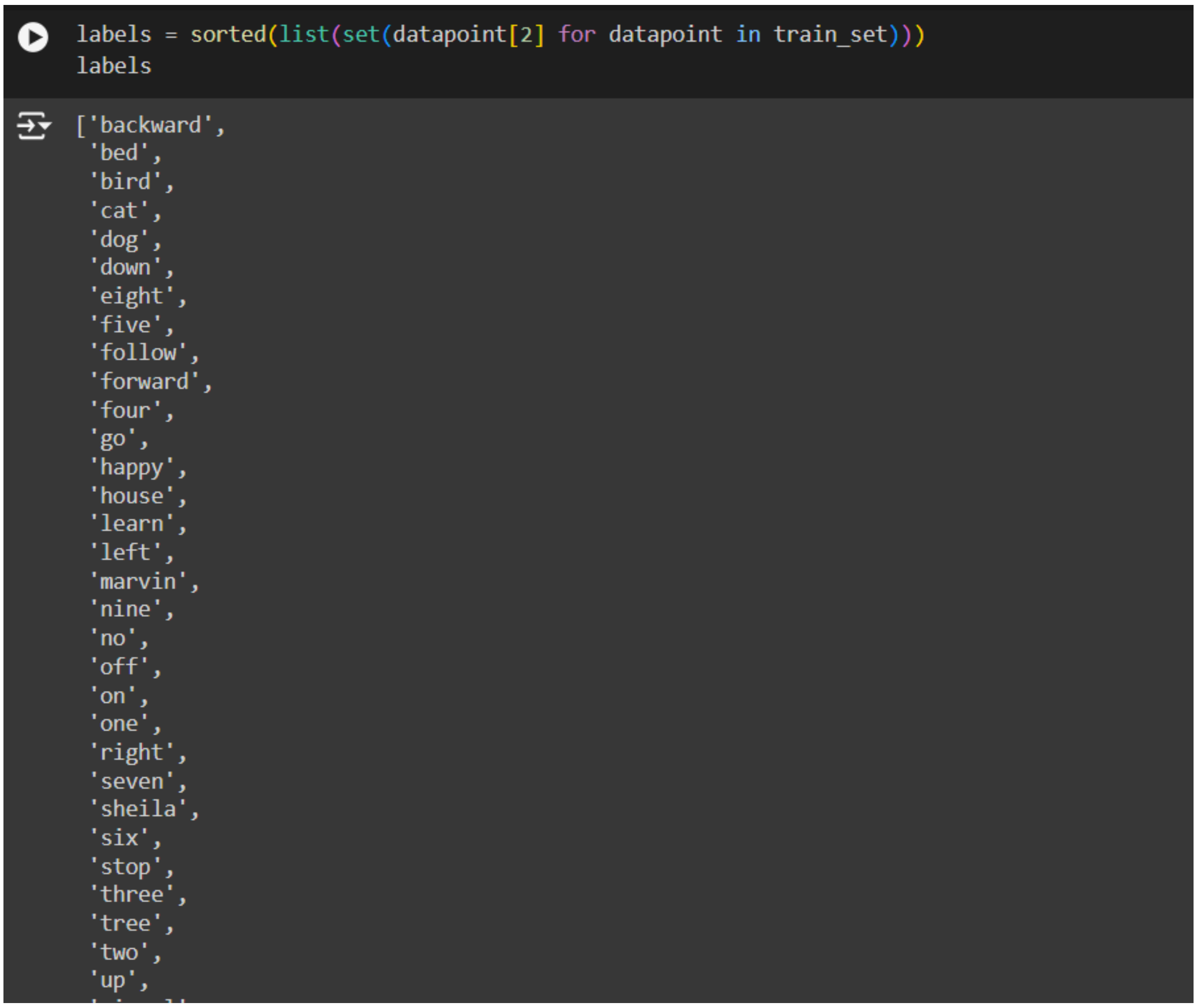
```
import rarfile
import os

def extract_rar(rar_path, extract_to):
    # Ensure the extract directory exists
    if not os.path.exists(extract_to):
        os.makedirs(extract_to)

    # Open and extract the .rar file
    with rarfile.RarFile(rar_path) as rf:
        rf.extractall(path=extract_to)

    print(f"Extracted to {extract_to}")

# Path to your .rar file and where to extract
rar_file_path = "/content/recordings_v2.rar"
extract_to_path = "/content"

# Extract the .rar file
extract_rar(rar_file_path, extract_to_path)
```

```
Extracted to /content
```



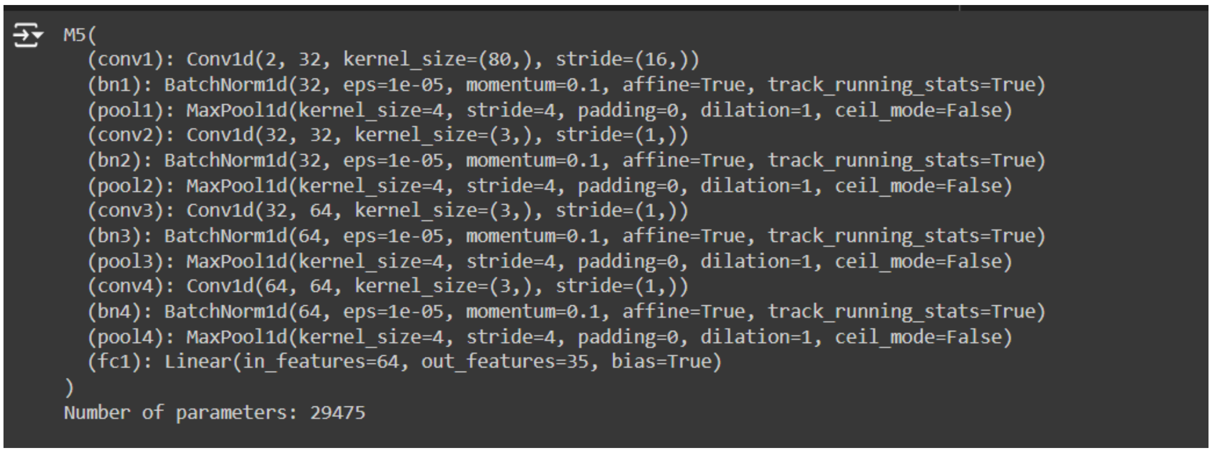
```
print("Shape of waveform: {}".format(waveform.size()))
print("Sample rate of waveform: {}".format(sample_rate))
print("Label of waveform: {}".format(label))

plt.plot(waveform.t().numpy());
```

```
Shape of waveform: torch.Size([2, 24235])
Sample rate of waveform: 16000
Label of waveform: backward
```

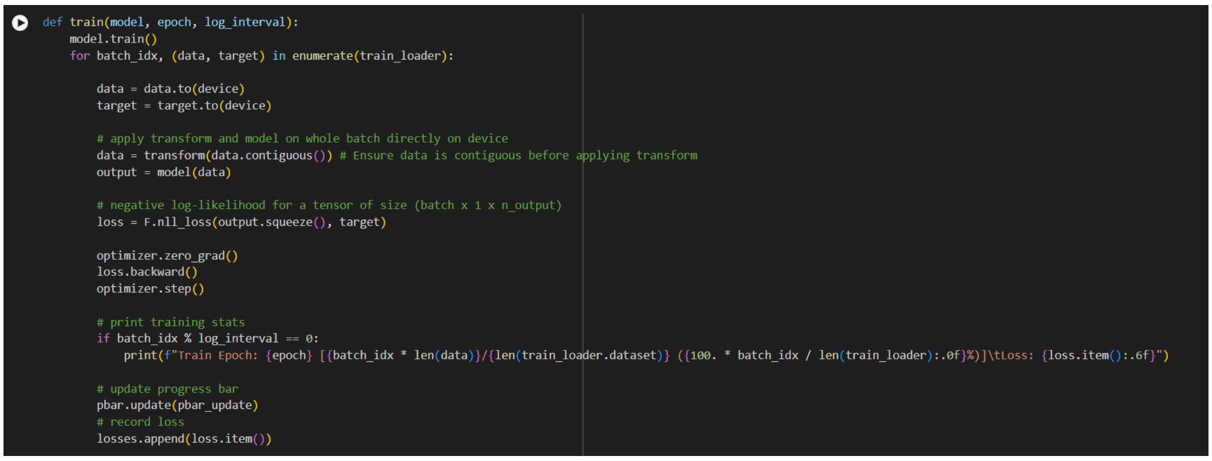


```
labels = sorted(list(set(datapoint[2] for datapoint in train_set)))
labels
```

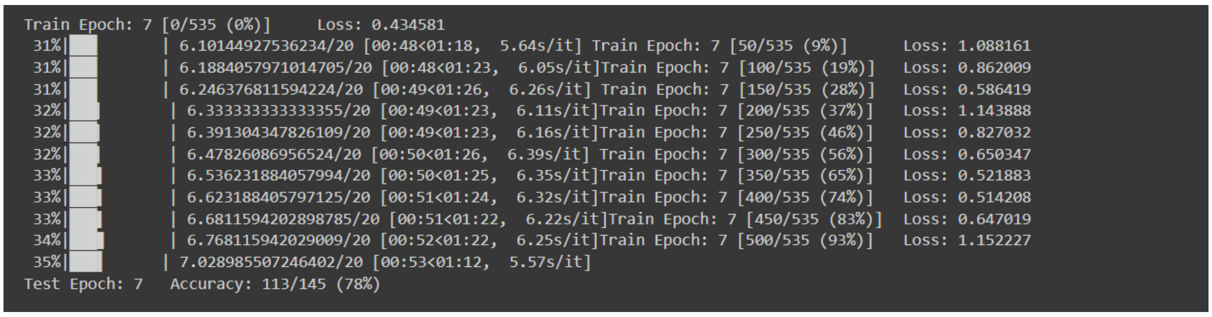
```
['backward',
 'bed',
 'bird',
 'cat',
 'dog',
 'down',
 'eight',
 'five',
 'follow',
 'forward',
 'four',
 'go',
 'happy',
 'house',
 'learn',
 'left',
 'marvin',
 'nine',
 'no',
 'off',
 'on',
 'one',
 'right',
 'seven',
 'sheila',
 'six',
 'stop',
 'three',
 'tree',
 'two',
 'up',
```



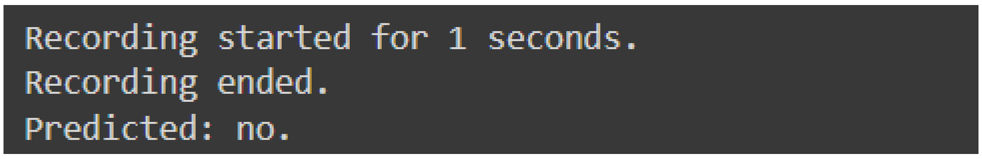
```
M5(
  (conv1): Conv1d(2, 32, kernel_size=(80,), stride=(16,))
  (bn1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(32, 32, kernel_size=(3,), stride=(1,))
  (bn2): 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: 29475
```



```python
def train(model, epoch, log_interval):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):

        data = data.to(device)
        target = target.to(device)

        # apply transform and model on whole batch directly on device
        data = transform(data.contiguous()) # Ensure data is contiguous before applying transform
        output = model(data)

        # negative log-likelihood for a tensor of size (batch x 1 x n_output)
        loss = F.nll_loss(output.squeeze(), target)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # print training stats
        if batch_idx % log_interval == 0:
            print(f"Train Epoch: {epoch} [{batch_idx * len(data)}/{len(train_loader.dataset)} ({100. * batch_idx / len(train_loader):.0f}%)]\tLoss: {loss.item():.6f}")

        # update progress bar
        pbar.update(pbar_update)
        # record loss
        losses.append(loss.item())
```



```
Train Epoch: 7 [0/535 (0%)]     Loss: 0.434581
 31%|███       | 6.10144927536234/20 [00:48<01:18,  5.64s/it] Train Epoch: 7 [50/535 (9%)]      Loss: 1.088161
 31%|███       | 6.1884057971014705/20 [00:48<01:23,  6.05s/it]Train Epoch: 7 [100/535 (19%)]   Loss: 0.862009
 31%|███       | 6.246376811594224/20 [00:49<01:26,  6.26s/it] Train Epoch: 7 [150/535 (28%)]   Loss: 0.586419
 32%|███▏      | 6.333333333333355/20 [00:49<01:23,  6.11s/it]Train Epoch: 7 [200/535 (37%)]   Loss: 1.143888
 32%|███▏      | 6.391304347826109/20 [00:49<01:23,  6.16s/it]Train Epoch: 7 [250/535 (46%)]   Loss: 0.827032
 32%|███▏      | 6.47826086956524/20 [00:50<01:26,  6.39s/it] Train Epoch: 7 [300/535 (56%)]   Loss: 0.650347
 33%|███▎      | 6.536231884057994/20 [00:50<01:25,  6.35s/it]Train Epoch: 7 [350/535 (65%)]   Loss: 0.521883
 33%|███▎      | 6.623188405797125/20 [00:51<01:24,  6.32s/it]Train Epoch: 7 [400/535 (74%)]   Loss: 0.514208
 33%|███▎      | 6.6811594202898785/20 [00:51<01:22,  6.22s/it]Train Epoch: 7 [450/535 (83%)]  Loss: 0.647019
 34%|███▍      | 6.768115942029009/20 [00:52<01:22,  6.25s/it]Train Epoch: 7 [500/535 (93%)]   Loss: 1.152227
 35%|███▌      | 7.028985507246402/20 [00:53<01:12,  5.57s/it]
Test Epoch: 7   Accuracy: 113/145 (78%)
```



```
Recording started for 1 seconds.
Recording ended.
Predicted: no.
```