

**RECOGNISE MY VOICE COMMANDS**  
**SPEECH PROCESSING AND SYNTHESIS**



**Name: Aditya Parmar**  
**Roll Number: 102103500**

**Submitted to:**  
**Dr. Raghav B. Venkataramaiyer**

**THAPAR INSTITUTE OF ENGINEERING AND  
TECHNOLOGY,**  
**(A DEEMED TO BE UNIVERSITY),**  
**PATIALA, PUNJAB**  
**INDIA**

## SUMMARY

The "Speech Commands Dataset," a set of spoken words for keyword detecting system training, is introduced in the paper. Effective models should be able to comprehend commands like "Yes," "No," and "Stop" because they are meant for low-resource devices. The dataset helps to improve model comparability and reproducibility. It contains baseline models with an accuracy of 88.2%. This collection consists of more than 100,000 words from 2,618 speakers.

## DATASET SUMMARY

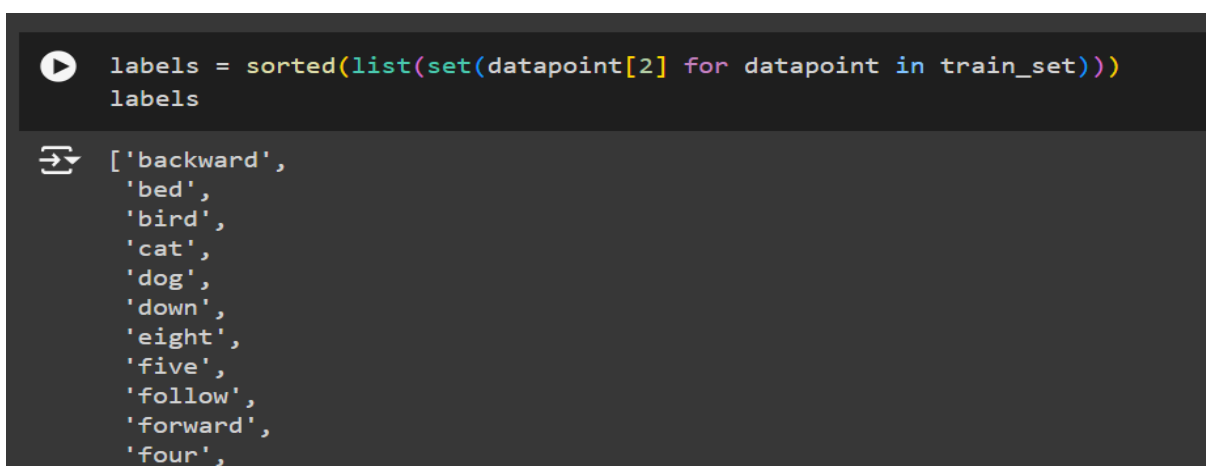
The study paper's Speech Commands Dataset includes over 105,829 audio recordings of 35 distinct words. Every audio file in the WAV format has a single spoken word that is sampled at 16 kHz. Since 2,618 speakers provided the dataset, a wide range of accents and pronunciations were guaranteed. With words like "Yes," "No," "Up," "Down," "Left," "Right," "On," "Off," "Stop," "Go," and other words like numerals (zero to nine), it primarily focuses on small-vocabulary keyword detecting tasks.

When uncompressed, the complete dataset is about 3.8 GB, or 2.7 GB when saved as a tar archive that has been compressed using gzip. It also comes with files including background noise to mimic actual situations and increase the robustness of the model.

## PERFORMANCE MEASURES

The accuracy of the model came out to be 82% after 18 epochs.

## CODE SNAPSNOTS



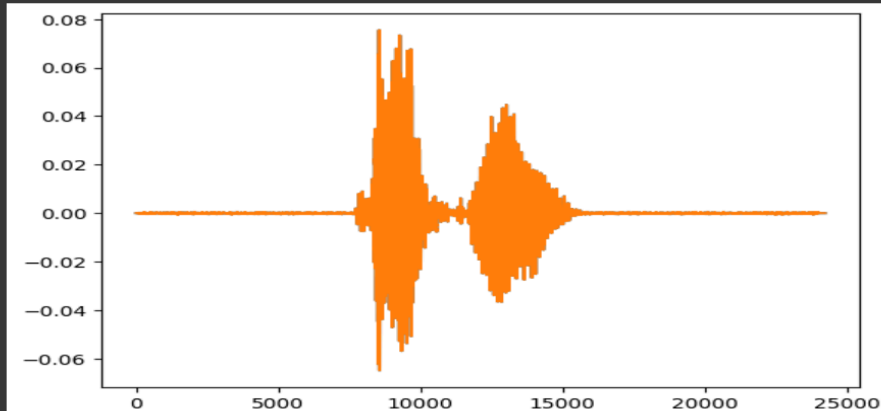
```
▶ labels = sorted(list(set(datapoint[2] for datapoint in train_set)))
labels

⇒ ['backward',
   'bed',
   'bird',
   'cat',
   'dog',
   'down',
   'eight',
   'five',
   'follow',
   'forward',
   'four',
```

```
[ ] 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
```



```
▶ log_interval = 5
   n_epoch = 22

   pbar_update = 1 / (len(train_loader) + len(test_loader))
   losses = []

   transform = transform.to(device)
   with tqdm(total=n_epoch) as pbar:
       for epoch in range(1, n_epoch + 1):
           train(model, epoch, log_interval)
           test(model, epoch)
           scheduler.step()
```

```
⇒ 0%|          | 0.028985507246376812/20 [00:00<02:24, 7.24s/it]Train Epoch: 1 [0/535 (0%)]    Loss: 3.744446
   1%|          | 0.11594202898550723/20 [00:00<02:15, 6.82s/it]Train Epoch: 1 [50/535 (9%)]    Loss: 3.252906
   1%|          | 0.1739130434782609/20 [00:01<02:17, 6.91s/it] Train Epoch: 1 [100/535 (19%)]   Loss: 3.254390
   1%|          | 0.26086956521739135/20 [00:01<02:10, 6.63s/it]Train Epoch: 1 [150/535 (28%)]   Loss: 3.637258
   2%|          | 0.318840579710145/20 [00:02<02:07, 6.46s/it] Train Epoch: 1 [200/535 (37%)]   Loss: 3.477170
   2%|          | 0.4057971014492755/20 [00:02<02:11, 6.71s/it] Train Epoch: 1 [250/535 (46%)]   Loss: 3.174408
   2%|          | 0.46376811594202916/20 [00:03<02:09, 6.62s/it]Train Epoch: 1 [300/535 (56%)]   Loss: 3.600039
   3%|          | 0.5217391304347827/20 [00:03<02:03, 6.34s/it]Train Epoch: 1 [350/535 (65%)]   Loss: 2.957685
   3%|          | 0.6086956521739129/20 [00:04<02:04, 6.42s/it]Train Epoch: 1 [400/535 (74%)]   Loss: 3.201174
   3%|          | 0.6811594202898547/20 [00:04<02:42, 8.39s/it]Train Epoch: 1 [450/535 (83%)]   Loss: 3.261173
   4%|          | 0.7536231884057965/20 [00:05<03:02, 9.48s/it]Train Epoch: 1 [500/535 (93%)]   Loss: 3.059032
   5%|          | 1.0289855072463754/20 [00:07<02:08, 6.77s/it]
   Test Epoch: 1 Accuracy: 6/145 (4%)

   Train Epoch: 2 [0/535 (0%)]    Loss: 3.222335
   6%|          | 1.1159420289855062/20 [00:08<02:03, 6.54s/it]Train Epoch: 2 [50/535 (9%)]    Loss: 3.050375
   6%|          | 1.17391304347826/20 [00:08<02:02, 6.52s/it] Train Epoch: 2 [100/535 (19%)]   Loss: 2.944656
   6%|          | 1.260869565217391/20 [00:09<02:05, 6.71s/it]Train Epoch: 2 [150/535 (28%)]   Loss: 3.452490
```

```

        b" recorder.start()\n"
        b" await sleep(time)\n"
        b" recorder.onstop = async ()=>\n"
        b"     blob = new Blob(chunks)\n"
        b"     text = await b2text(blob)\n"
        b"     resolve(text)\n"
        b" }\n"
        b" recorder.stop()\n"
        b"})"
    )
    RECORD = RECORD.decode("ascii")

    print(f"Recording started for {seconds} seconds.")
    display(ipd.Javascript(RECORD))
    s = colab_output.eval_js("record(%d)" % (seconds * 1000))
    print("Recording ended.")
    b = b64decode(s.split(",")[1])

    fileformat = "wav"
    filename = f"_audio.{fileformat}"
    AudioSegment.from_file(BytesIO(b)).export(filename, format=fileformat)
    return torchaudio.load(filename)

# Detect whether notebook runs in google colab
if "google.colab" in sys.modules:
    waveform, sample_rate = record()
    print(f"Predicted: {predict(waveform)}.")
    ipd.Audio(waveform.numpy(), rate=sample_rate)

```

```

# Detect whether notebook runs in google colab
if "google.colab" in sys.modules:
    waveform, sample_rate = record()
    print(f"Predicted: {predict(waveform)}.")
    ipd.Audio(waveform.numpy(), rate=sample_rate)

```

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