Speech Command Model

SENANI SADHU

February 13, 2021

1 Language Used:

Python

2 Libraries and packages used:

KAPRE , SCIKIT LEARN , SOUND FILE , TENSORFLOW.

3 Overview:

Speech Command Recognition is changing voices in to text form,Speech command recognition is present in a wide range of devices and utilized by many HCI interfaces. In many situations, it is desirable to obtain lightweight and high accuracy models that can run locally. Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated vocabulary into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker independent"[1] systems. Systems that use training are called "speaker dependent". Speech recognition applications include voice user interfaces such as voice dialing (e.g. "call home"), call routing (e.g. "I would like to make a collect call"), domotic appliance control, search key words (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), determining speaker characteristics,[2] speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

4 DATA:

- Set 16KHz as sampling rate.
- Record 80 utterances of each command.
- Save samples of each command in different folders.
    1. Dataset/Forward.
    2. Dataset/Back.
    3. Dataset/Left.

4. Dataset/Right.
5. Dataset/Stop.

5 Description:-

1. Using Convolutional layers ahead of LSTM is shown to improve performance in several research papers.
2. BatchNormalization layers are added to improve convergence rate.
3. Using Bidirectional LSTM is optimal when complete input is available. But this increases the runtime two-fold.
4. Final output sequence of LSTM layer is used to calculate importance of units in LSTM using a FC layer.
5. Then take the dot product of unit importance and output sequences of LSTM to get Attention scores of each time step.
6. Take the dot product of Attention scores and the output sequences of LSTM to get attention vector.
7. Add an additional FC Layer and then to output Layer with SoftMax Activation.

6 RUN:

The Code is written using Google Colab:

1. Open ColabNotebook.ipynb and change Runtime to GPU.
2. Upload Speech-Recognition-Project/Data to Colab.
3. Run the cells in the same order in Notebook Test.

7 TEST:

1. Locate the folder where you save your model.h5 file.
2. Start speaking when you see mike in the bottom right plane of the task bar or see red blinking dot in the title bar.

8 MODEL SUMMARY:

```
model.summary()
[15]

Model: "Attention"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Input (InputLayer)              [(None, 49, 39, 1)]  0
__________________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 49, 39, 10)   60          Input[0][0]
__________________________________________________________________________________________________
BN1 (BatchNormalization)        (None, 49, 39, 10)   40          Conv1[0][0]
__________________________________________________________________________________________________
Conv2 (Conv2D)                  (None, 49, 39, 1)    51          BN1[0][0]
__________________________________________________________________________________________________
BN2 (BatchNormalization)        (None, 49, 39, 1)    4           Conv2[0][0]
__________________________________________________________________________________________________
Squeeze (Reshape)               (None, 49, 39)       0           BN2[0][0]
__________________________________________________________________________________________________
LSTM_Sequences (LSTM)           (None, 49, 64)       26624       Squeeze[0][0]
__________________________________________________________________________________________________
FinalSequence (Lambda)          (None, 64)           0           LSTM_Sequences[0][0]
__________________________________________________________________________________________________
UnitImportance (Dense)          (None, 64)           4160        FinalSequence[0][0]
__________________________________________________________________________________________________
AttentionScores (Dot)           (None, 49)           0           UnitImportance[0][0]
                                                                 LSTM_Sequences[0][0]
__________________________________________________________________________________________________
AttentionSoftmax (Softmax)      (None, 49)           0           AttentionScores[0][0]
__________________________________________________________________________________________________
AttentionVector (Dot)           (None, 64)           0           AttentionSoftmax[0][0]
                                                                 LSTM_Sequences[0][0]
__________________________________________________________________________________________________
FC (Dense)                      (None, 32)           2080        AttentionVector[0][0]
__________________________________________________________________________________________________
Output (Dense)                  (None, 5)            165         FC[0][0]
==================================================================================================
Total params: 33,184
Trainable params: 33,162
Non-trainable params: 22
__________________________________________________________________________________________________
```

Figure 1: Fig

9 The model is accomplished with an maximum accuracy of 97.8%.