Speech Command Recognition

Presentation by
P Venkatesh
AI20MTECH01004

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

September 26, 2021

Overview

Problem Statement

- The goal of the project is to develop a speech command recognition model
- keras's neural attention network is used as a baseline framework
- The model is trained and tested on speech command dataset to recognize 5 different commands as follows:
  - back
  - forward
  - left
  - right
  - stop

Model Architecture

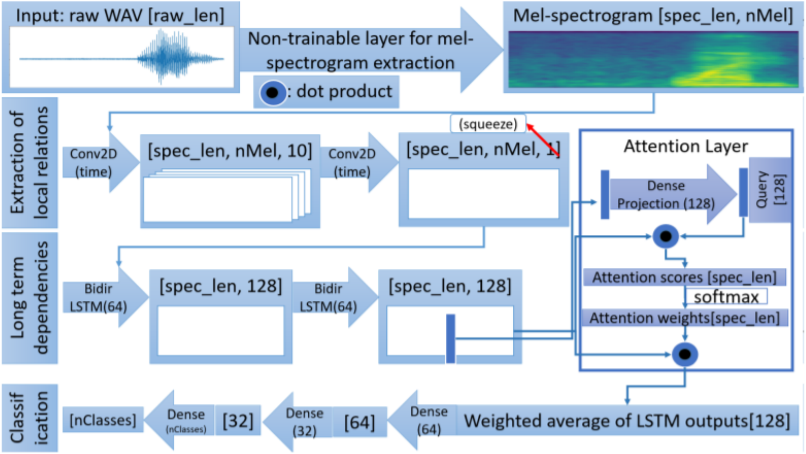
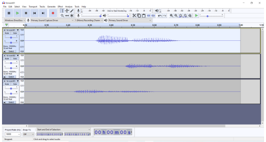


Figure 2: Attention Model

Dataset Preparation



(a) Dataset Preparation using Audacity

- Set 16KHz as sampling rate
- Record 80 utterances of each command
- Save samples of each command in different folders
  - Data/back
  - Data/forward
  - Data/left
  - Data/right
  - Data/stop

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 4: Neural Attention Network Architecture

Experiments and Results (cont.)

```
Epoch 1/10
510/510 - 15s - loss: 0.6114 - sparse_categorical_accuracy: 0.7829 - val_loss: 0.2484 - val_sparse_categorical_accuracy: 0.9112
Epoch 2/10
510/510 - 10s - loss: 0.0569 - sparse_categorical_accuracy: 0.9849 - val_loss: 0.0898 - val_sparse_categorical_accuracy: 0.9618
Epoch 3/10
510/510 - 10s - loss: 0.0171 - sparse_categorical_accuracy: 0.9960 - val_loss: 0.1021 - val_sparse_categorical_accuracy: 0.9647
Epoch 4/10
510/510 - 10s - loss: 0.0027 - sparse_categorical_accuracy: 0.9995 - val_loss: 0.1050 - val_sparse_categorical_accuracy: 0.9597
Epoch 5/10
510/510 - 10s - loss: 5.4265e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.1070 - val_sparse_categorical_accuracy: 0.9629
Epoch 6/10
510/510 - 10s - loss: 2.8190e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0999 - val_sparse_categorical_accuracy: 0.9659
Epoch 7/10
510/510 - 10s - loss: 1.8118e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0991 - val_sparse_categorical_accuracy: 0.9653
Epoch 8/10
510/510 - 10s - loss: 1.2395e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.1049 - val_sparse_categorical_accuracy: 0.9676
Epoch 9/10
510/510 - 10s - loss: 8.5585e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.1056 - val_sparse_categorical_accuracy: 0.9700
Epoch 10/10
510/510 - 10s - loss: 6.0929e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.1074 - val_sparse_categorical_accuracy: 0.9703
<keras.callbacks.History at 0x7fc36019abd0>
```

Figure 5: Results

Conclusion

- The final accuracy of model is 0.97. Even with this high accuracy the test commands on model give false positives. Hence the model overfits the data.
- We attribute overfitting to the fact that the data is less and each class has similar data making it hard for the model to generalize.

THANK YOU!