ML4IOT HomeWork2

Authors : Azimkhan Orazalin, Ali Abbas Syed, Bilal Shabbir
Studentid : s298255, s292423, s305979
Politecnico di Torino

***Abstract*—In the first part of HW2 Go/stop Classifier model was build, trained and was optimised using the best Hyper-parameters for Pre-Processing and Training. The second part is battery monitoring script based on VAD and tflite model to start and stop uploading the battery power and battery plugging values to Redis timeseries.**

I. First task

Basic model architecture used during the lab was used and fine-tuned for purpose of using it as GO/Stop classifier model. Data contained only the files of Go/stop wav format from the MSC data set was used . Pre-processing file was used to defined the functions needed in the Pre processing phase of audio signal.**Get-audio-label** and **get-spectogram** was used as they are from the last HW the function.**Get-log-mel-spectogram** defined in preprocessing file that returns the Mfcss of signal.For MFCSS get-mfccs function was not used.After this model was defined and Grid search was performed to get best hyper-parameters for pre-processing from a range of arguments and for value of alpha for width scaling.Parameters range was used as defined during the lab and parameters for final model are reported in table.Training arguments was used as they were given during the lab.

The hyper-parameters for which we Achieved the **Accuracy $>$ 97 percent** and **TFLite Size $<$ 25KB** and **Total Latency $<$ 8ms** are reported in the Figure:

Hyper-parameters of the final solution	
Hyper-parameter	values
downsampling_rate	16000
frame_length_in_s	0.016
frame_step_in_s	0.012
num_mel_bin	10
lower_frequency	20
upper_frequency	8000
num_coefficients	40
alpha	0.25

Training arguments	
Hyper-parameter	values
batch_size	20
initial_learning_rate	0.01
end_learning_rate	1.e-5
epochs	10

DS-CNN model architecture having layers [Input(shape),Conv2D(), BatchNormalization(),ReLU(),DepthwiseConv2D(),Conv2D(),BatchNormalization() ,ReLU(),DepthwiseConv2D(),Conv2D(),BatchNormalization(), ReLU(),GlobalAveragePooling2D(),Dense(),Softmax()].
For all 2D layers filter of size 128 or 256 was used alpha as scaling factor and kernal size of [3*3] with stride of [1*1] was used except first 2D layer where stride of [2*2] was used.

For optimization 128 and 256 were used with different scaling factor.256 filters achieved accuracy higher than 97% the size of trained model was bigger than 25kb.128 filters with alpha=0.25 fulfilled both constraints.

Final Results	
Variables	Values
Accuracy	99
Latency	6.6ms
TFLite size	15.9

II. Second task

A. *Updating the battery monitoring script*

In this task was integrated Voice User interface based on **VAD and KWS**. Initially the script always goes in background and records voices around laptop's microphone. To classify the labels and noise was used **the best and optimized tflite model** described above. Main goal is to store data in Redis cloud with using speech recognition model, **"go" and "stop"** voice commands. Continuously going script tries to catch our label words. If model recognize a speech, it runs a model, and predicts keyword. **Threshold accuracy** for both keywords is **95 percent**.

1) Script firstly check if there is speech around microphone.
2) If model recognize go with **more than 95 percent**, it starts to store **plugged_seconds** and **battery_power** into the Redis cloud, and let us to know by printing **"Started to add in Redis"**
3) If model recognize stop with the same threshold. We will get **"Stopped to add in Redis"** notification and stop data storing script.
4) In the case if script does not reach the accuracy, it stays in a current state.

While script running. We always can exit from the continuously recording by pressing **Q** button.