# Lab 7 - TensorFlow for voice recognition

TensorFlow is a library for different data manipulations, but it is mostly used for neural networks. TensorFlow Lite is its light-weight counterpart suitable for running on Android devices. In this lab we will first see:

1. how to record audio using android devices;
2. how to use a pre-trained neural network model for real-time voice recognition;
3. how to dynamically adjust the sensing rate.

## Starting application

To shortcut the development process, you can download the starting code from:

https://bitbucket.org/pbdfrita/msfrita_voice_recognition

The application is based on the TensorFlow Lite Speech Command Recognition Android Demo. You should be able to run the application on your Android device (or emulator). The application performs recognition of speech commands and highlights the recognized word. Test it and make sure that it works.

It is a simple application that contains only two activities:

1. **RecognizeActivity.java** - helper activity that simplifies the recognition process;
2. **SpeechActivity.java** - main activity which implements all functionalities. We will focus on this activity.

Check the **onCreate()** method and try to understand the six functions called from there:

1. **loadRecognitionSettings()** - Sets recognition parameters;
2. **loadModel()** - Loads the neural network model for recognition of speech commands;
3. **requestMicrophonePermission()** - Checks permissions;
4. **startRecording()** - Starts sensing audio using the device's microphone;
5. **startRecognition()** - Applies the model on the sensed audio;
6. **setUI()** - Updates the user interface.

To make sure that the application works properly when being paused, you should call **startBackgroundThread()** in the **onResume();** and call **stopBackgroundThread()** in the **onStop().** Make sure that you understand the two functions. How many Threads are used in the application?

## Layout update

At the bottom of the layout **activity_speech.xml** there is a TextView that shows the inference time. Bellow this TextView, add two more TextViews:

1. Detected Voice Activity: Shows the value of the variable ***VAD_SCORE***
2. Time Between Samples: Shows the value of the variable ***TIME_BETWEEN_SAMPLES_MS***

## Voice Recognition

The results of the most recent recognition process can be accessed via the object RecognizeCommands.RecognitionResult ***result*** in the function **recognize().** The value ***result.score*** represents an estimation about how certain the model is that a specific word has been recognized. The higher this score is, the more certain the model is. Use this score to create a module for Voice Activation Detection (VAD). You should:

- calculate the ***VAD_SCORE*** using the equation: ***VAD_SCORE =*** 100 * result.**score**;
- update the function **checkVAD**( ***VAD_SCORE***) so that it:

    1. updates the UI (it shows the latest value of the ***VAD_SCORE***)
    2. changes the UI background if a voice has been detected (***VAD_SCORE*** > ***VAD_THRESHOLD).*** If not, it should stay white.

- test the application. Is your VAD module working i.e., is the background color changing when you speak close to the microphone?

## Dynamically adjust the sensing rate

Currently, the application senses at a constant rate, which can be changed using the variable ***TIME_BETWEEN_SAMPLES_MS*****.** This variable tells the application how long the recording thread should sleep, before it records the next audio sample.

Now that we have a working VAD module, we can use it to dynamically adjust the sensing rate. If there is no voice detected, the device should sense less often. If there is a voice detected, the device should sense more often. Update the application using the following guidelines:
- If a voice has been detected, the sensing rate should increase for 50%;
- If a voice has NOT been detected, the sensing rate should decrease for 50%;
- The sensing rate should not be smaller than ***MINIMUM_TIME_BETWEEN_SAMPLES_MS;***
- The sensing rate should not be larger than ***MAX_TIME_BETWEEN_SAMPLES_MS;***
- Update the TextView Time_between_samples, every time the sensing rate changes.

If you did not attend the lab slot in-person at FRI, you should commit your solution to a private Bitbucket repository named **FRIMS2021-LAB-7** and a user **pbdfrita** (**pbdfrita@gmail.com**) should be added as a read-only member. The solutions will be pulled from your repository on **Sunday, May 8th, 23:59**.