Recording New Data

Design “start recording” as a button to start , and design an EditText to input a new gesture name you want to simulate, then basically it is the same as the input data of this assignment. In the onSensorChange module, the detected data use a low-pass filter to Isolate the force of gravity and remove the gravity contribution with the high-pass filter, and multiply by 1000 to convert to the complete_data format. Still use nsamples to ensure that 128 data points are still collected.
The type of data at this time is Float and needs to be converted to String by using .toString().The format is as follows:
{"gesture": "which type of gesture", "accel_xyz": [[, linear_acceleration[0], linear_acceleration[1],linear_acceleration[2]],

After the end of an experiment, according to the sample app “FileReadWriteDemo”, use the write_file_internal function to write the current data as a row to txt file. Save in cache / files / <gesture_name> _output.txt. Multiple experiments are stored in different rows. Make sure that one text contains data for one person performing a single gesture. Different gestures use different txt to storage.
It is worth noting that, make sure that the number of samples of different gestures are the same,128. If the sample length is inconsistent, data normalization will be required.

I also need to provide data for the "unknown" category, naming “negative”. This will be trained together with the previous “unknown” data.

Modifying in the train.py file

I need to modify the Last dense layer. Because a new gesture is added, the number of neurons in this layer is 3.

Changes in the data_split.py file

First in data_load.py, self.label2id = {"wing": 0, "negative": 1} needs to command the new gesture added. For example, self.label3id = {"wing": 0, "negative": 1, “circle”: 2}. Correspondingly, in data_split, add a new gesture to num_dic = {"wing": 0, "negative": 0, “circle”: 0} to count.

The point that can be optimized is to perform random splits instead of fixed 0.6,0.2. It can improve the stability of the model.

Training the new model

First, I need to add the new generated output.txt to the data folder. Next, tune model hyperparameters. The parameters that can be modified and tried are number of layers, number of neurons, window size, dropouts, the number of epochs.
Notice: The number of columns in the window size should be 3.
Through TensorBoard comprehensive evaluation of the speed of model convergence and the combination of accuracy, choose acceptable and relatively good parameter combinations and output the new model.

Change in the Android App

Since I add a new gesture, OUTPUT_CLASSES_COUNT = 3.
Val result = Array(size:1){ FLoatArray(OUTPUT_CLASSES_COUNT)}