Fall Detection Software on Android

This document explains a fall detection software that is developed in Android. The software detects when falls happen by using a deep learning model. The deep learning model was trained by using a simplified version of the UMAFall Dataset[1]. When a fall is detected, the software sends patient information and location data to the server and also has the ability to send SMS.

Collecting Sensor Data on an Android Device

Most Android smartphones have built-in sensors (accelerometer and gyroscope) that measure motion and orientation. These sensors are capable of measuring and providing data with high precision and accuracy so they are useful to monitor three-dimensional device movement or positioning. For this work, built-in accelerometer and gyroscope data were used to predict the falling.

Saving measured data to built-in SQLite database

Accelerometer and gyroscope data were saved in a built-in SQLite database.

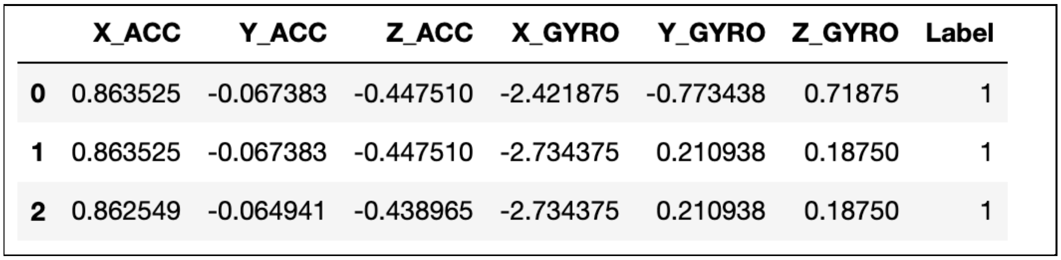
Deep Learning model and using it on Android programming

A simplified version of the UMAFall Dataset was used for training. Original UMAFall Dataset contains 746 csv files which belong to 19 participants, the size of the files is 373.88 GB. 3 individuals (Subject-2, Subject-4 and Subject-14) and 4 activities (Walking, BackwardFall, ForwardFall and LateralFall) were chosen for the sake of simplicity. Details of selected subjects are below:

Subject	Gender, Age	Activities	Device
2	Female, 22	● Walking ● BackwardFal ● ForwardFall ● LateralFall	LG G4 and sensor tag
4	Male, 27	● Walking ● BackwardFal ● ForwardFall ● LateralFall	LG G4 and sensor tag
14	Female, 22	● Walking ● BackwardFal ● ForwardFall ● LateralFall	Samsung S5 and sensor tag

In the original UMAFall work, smartphone gyroscope data wasn’t included so smartphone accelerometer and waist sensor tag was used for gyroscope data.

Processed data looks like below:



	X_ACC	Y_ACC	Z_ACC	X_GYRO	Y_GYRO	Z_GYRO	Label
0	0.863525	-0.067383	-0.447510	-2.421875	-0.773438	0.71875	1
1	0.863525	-0.067383	-0.447510	-2.734375	0.210938	0.18750	1
2	0.862549	-0.064941	-0.438965	-2.734375	0.210938	0.18750	1

Deep learning model is pretty simple and has only one hidden layer. Model trained 500 epoch and the batch size is 128. Model saved as TensorFlow Lite to be used in Android. Training code and the dataset can be seen in the Github repository[2]

```
 Layer (type)            Output Shape           Param #
=================================================================
 dense_6 (Dense)          (None, 64)             448

 dense_7 (Dense)          (None, 1)              65
=================================================================
Total params: 513
Trainable params: 513
Non-trainable params: 0
```

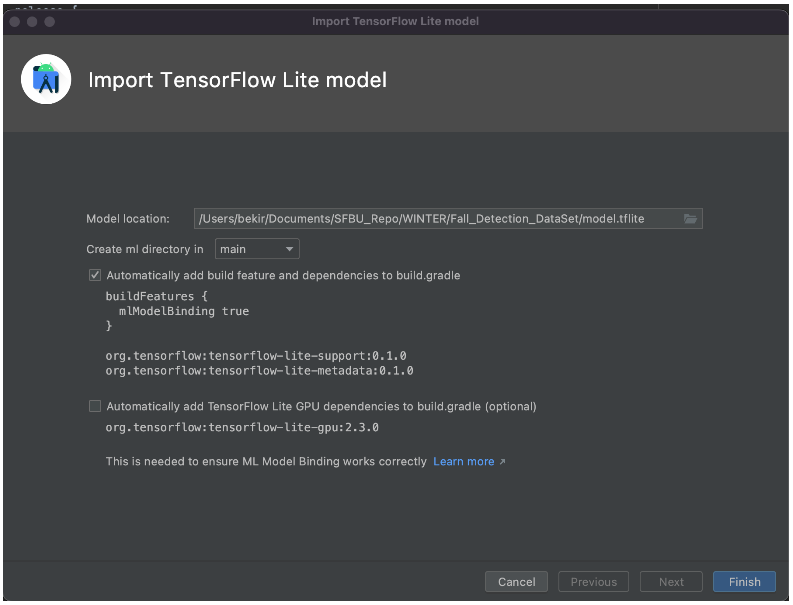
Using The Model in Android App

Trained model saved as TensorFlow Lite model to be used in Android. TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices[3].

Here is the process of using the TFlite model in an Android App:

Right-click on the app file the TFLite model or click on File, then New > Other > TensorFlow Lite Model

Select the location of your TFLite file. After this process, Android Studio will configure the module's dependency on your behalf with ML Model binding and all dependencies automatically inserted into your Android module's build.gradle file. The process is simple as below:



Import TensorFlow Lite model

Model location: /Users/bekir/Documents/SFBU_Repo/WINTER/Fall_Detection_DataSet/model.tflite

Create ml directory in: main

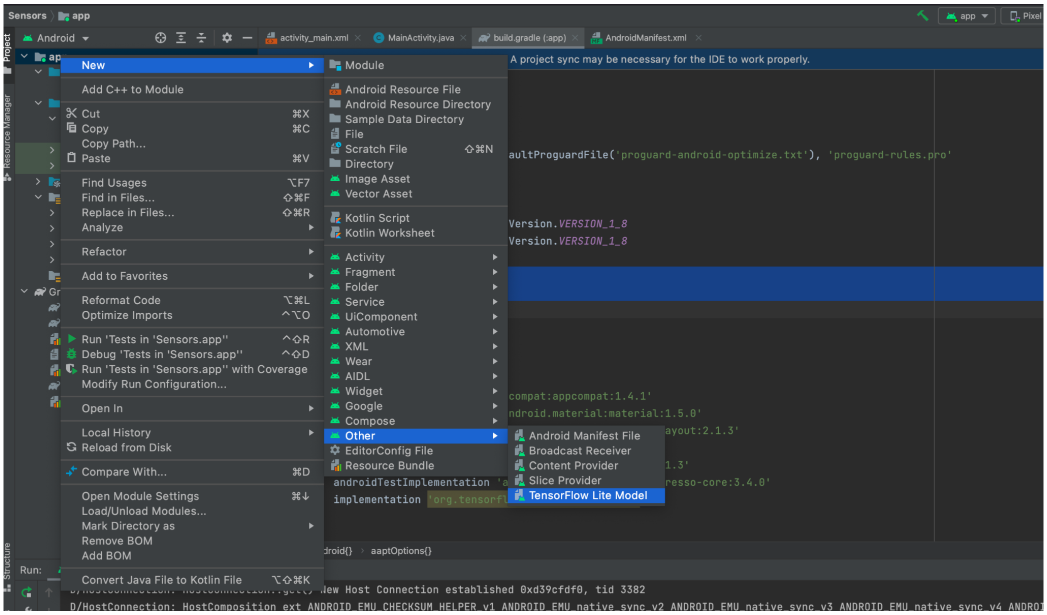
☑ Automatically add build feature and dependencies to build.gradle

```
buildFeatures {
  mlModelBinding true
}

org.tensorflow:tensorflow-lite-support:0.1.0
org.tensorflow:tensorflow-lite-metadata:0.1.0
```

☐ Automatically add TensorFlow Lite GPU dependencies to build.gradle (optional)

```
org.tensorflow:tensorflow-lite-gpu:2.3.0
```

This is needed to ensure ML Model Binding works correctly Learn more ↗

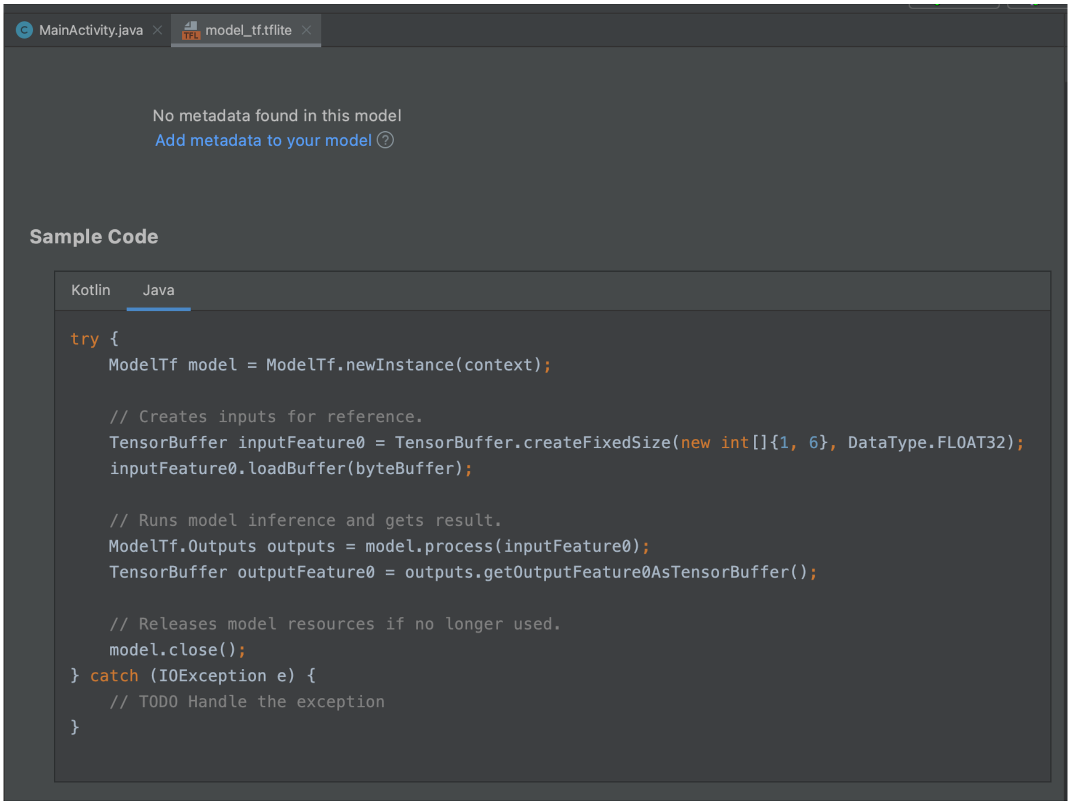


A project sync may be necessary for the IDE to work properly.

```
aultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
Version.VERSION_1_8
Version.VERSION_1_8
compat:appcompat:1.4.1'
ndroid.material:material:1.5.0'
ayout:2.1.3'
1.3'
resso-core:3.4.0'
androidTestImplementation 'a
implementation 'org.tensorfl
droid{} › aaptOptions{}
```

```
New Host Connection established 0xd39cfdf0, tid 3382
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROI
```

After adding the tflite model, we can start to get predictions from the model. Android Studio gives sample code like below to show how to use the model but we need to install tflite-support `pip install tflite-support` to get the sample code like below. The model gets a float array of 6 float numbers and gives a prediction.



No metadata found in this model

Add metadata to your model

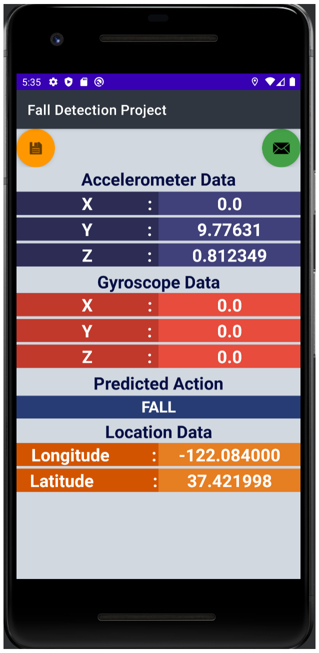
Sample Code

```
try {
    ModelTf model = ModelTf.newInstance(context);

    // Creates inputs for reference.
    TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 6}, DataType.FLOAT32);
    inputFeature0.loadBuffer(byteBuffer);

    // Runs model inference and gets result.
    ModelTf.Outputs outputs = model.process(inputFeature0);
    TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

    // Releases model resources if no longer used.
    model.close();
} catch (IOException e) {
    // TODO Handle the exception
}
```

Sending data to server by using TCP programming and SMS function

When a fall is detected, the program sends the patient and location data to the server and an sms to a provided phone number. The Java server program can be seen in the Github repo.

UI of the program



Fall Detection Project

Accelerometer Data

X	:	0.0
Y	:	9.77631
Z	:	0.812349

Gyroscope Data

X	:	0.0
Y	:	0.0
Z	:	0.0

Predicted Action

FALL

Location Data

Longitude	:	-122.084000
Latitude	:	37.421998

Sources

[1] http://webpersonal.uma.es/de/ECASILARI/Fall_ADL_Traces/UMA_FALL_ADL_dataset.html

[2] https://github.com/inputvector/Simple-Deep-Learning-Model-for-Fall-Detection

[3] https://www.tensorflow.org/lite/guide/android