**TENGINE**

**REAL-TIME**

**OBJECT DETECTION**

OPEN AI LAB

2019.7.15

Contents

[INTRODUCTION 3](#_Toc14094840)

[ARCHITECTURE 3](#_Toc14094841)

[Camera 3](#_Toc14094842)

[Image preprocess 4](#_Toc14094843)

[Tengine 4](#_Toc14094844)

[INSTALLATION 4](#_Toc14094845)

[System and Android Studio 4](#_Toc14094846)

[Optional 5](#_Toc14094847)

[Build your own library 6](#_Toc14094848)

[DEMO 6](#_Toc14094849)

# INTRODUCTION

This app is built based on [**Tengine**](https://github.com/OAID/Tengine), developed by **OPEN** AI LAB, which is a lite, high-performance, and modular inference engine for embedded device. The app is the product of machine learning and deep neural networks, and includes the content to help users define the object in front of them. Thus, it is general enough to be applicable in wide variety for future expansion.

In this project, we can use Tengine to run most popular models such as TensorFlow Mobilenet.v1/2 pb model, MobilenetSSD Caffe model and int8 TFLite model to predict image.

* This App supports for real-time camera image classification. It is built on previous project: [Tengine APP](https://github.com/OAID/Tengine-app/tree/master/android/classification)
* The App processes the real time image camera preview and gives the predict results.

# ARCHITECTURE

This app is developed in Android Studio. Main activity in Java is focused on Camera and data gathering. Then the data will be processed through JNI (Java Native Interface). JNI library is used to utilize the Tengine libraries and corresponding C APIs. Basically JNI provides the possible interaction between Tengine and our app.

## Camera

We use surfaceView to obtain the image from camera. After the moment user opens the app, it needs to initialize the camera event by adding surface callback. The surface callback has three main functions: surface created function, surface changed function and surface destroyed function.

During surface created function, it needs to check and find an available camera (front or back according to ID). Then it is ready to start the preview. Start preview function will adjust and optimize the interface based on specific device’s parameters (width and height etc.)

In the surface changed function, the preview will be autofocused when the preview changes. This is also the function where we get the preview image data.

In the last surface destroyed function, we simply stop preview and release camera when the camera is no longer in use.

## Image preprocess

After we get the image data in surface changed function, we will need to first rotate the image by 90 degrees clockwise because android devices originally obtain image by 90 degrees anti-clockwise by default. Then we convert the image into bitmap for Tengine to process.

## Tengine

*Init Tengine*: Initializes the Tengine every time it starts a new event. Among initialization, we define the predict model and its format and corresponding label file. Then it will create the graph based on selection. Last step in initialization is to pre-run the graph for tensor shape inference and make sure the environment is ready for a formal run.

*Get input data*: Resize the image based on the model input (e.g. 224\*224 for image size for MobileNet SSD Caffe model. Then manipulate the resized the image for future prediction.

*Run Tengine*: After gathering processed data, set the tensor buffer and run graph.

*Get top1*: Use Tengine to obtain couple possible results and stored in an array. Then we get the top score result and compare it to the label file to find the name of object.

# INSTALLATION

## System and Android Studio

Install the Android Studio, go to [Download](https://developer.android.google.cn/studio/). During installation, install the recommend tools and package as much as possible except the cloud ones.

Make sure your devices are based on Armv8.

You need to push your model by using adb. You should find it if you already installed Android Studio. It is located under Android\Sdk\platform-tools (Linux) and \AppData\Local\Android\Sdk\platform-tools(Windows). If you have not installed yet, use

For Linux:

sudo apt-get install android-tools-adb

For Windows:

If you already have Android SDK, it is under directory \sdk\platform-tools. Or download SDK here: <http://developer.android.com/sdk/index.html>

Find the adb and add it to your system environment variable. Then you can use adb command in terminal.

Use adb –version to make sure it is installed correctly.

Under the app/src/main/assets folder, place the .so libraries into app/src/main/JniLibs/arm64-v8a.

Use adb to push your model and label file in your file.

adb push your-model-name /data/local/tmp

In my case,

adb push mobilenet.caffemodel /data/local/tmp

Go to Android Studio, sync and build/run the project.

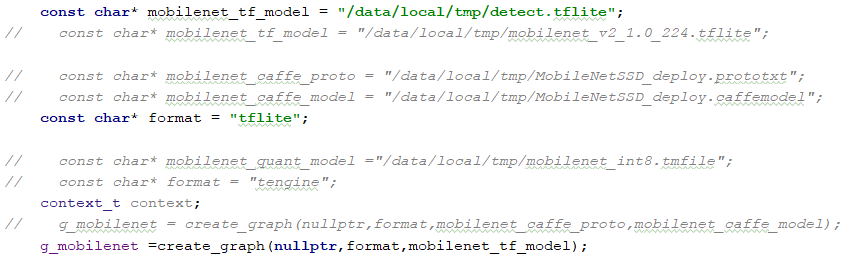
[https://github.com/OAID/Tengine-app/raw/master/android/classification/app/src/asset/Sync.png](https://github.com/OAID/Tengine-app/blob/master/android/classification/app/src/asset/Sync.png)

Also, make sure the App is given permission of using camera and SD card from your phone.

## Optional

You can see all the supported models here: [**Tengine model zoo**](https://pan.baidu.com/s/1LXZ8vOdyOo50IXS0CUPp8g#list/path=%2F) **(psw:57vb)**

You may need to edit the model and its label based on your preference.

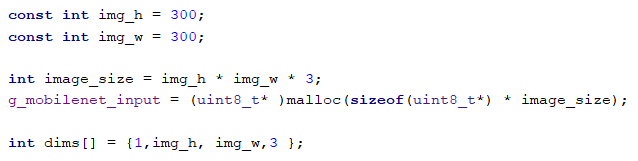


C:\Users\Lenovo\AppData\Local\Temp\1563168574.png

You need to change the corresponding model input and output.

e. g: In the following case, the input is 300\*300 for int8 TFLite model. And dimension is {1, 300, 300, 3}

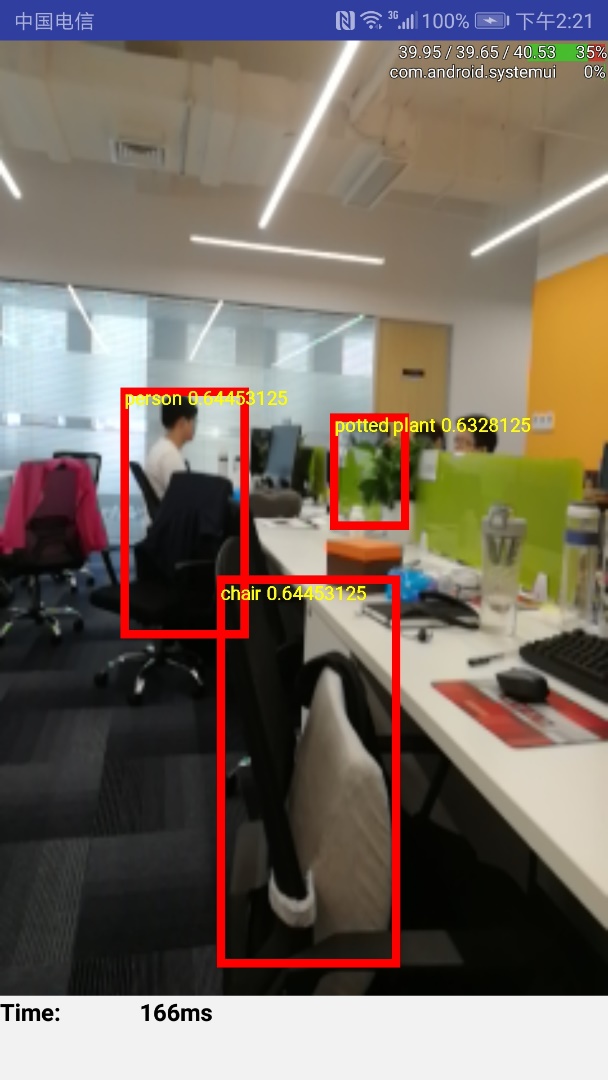
If you want to change model to MobileNet SSD Caffe model, the size then needs to be 224\*224. And dimension will be changed to {1, 3, 224, 224}. (Hint: Make sure you change all the code mentioning size and dimension in Tengine-Wrapper. The following snapshot is only for reference)



## Build your own library

If for some reason, you want to build your own Tengine library, it is ok. Please follow [**Android build**](https://github.com/OAID/Tengine/blob/master/doc/build_android.md). You can change the configure setting in CMakelist.

# DEMO



Predict results, corresponding labels and predict time are shown.