Ameema Zainab Follow
In love with Data. Data Enthusiast. Data Analytics Professional.
Jul 27 · 9 min read

Real-time Object Detection on Android using Tensorflow

Overview

Research shows that the detection of objects like a human eye has not been achieved with high accuracy using cameras and cameras cannot be replaced with a human eye. Detection refers to identification of an object or a person by training a model by itself. Detection of images or moving objects have been highly worked upon, and has been integrated and used in commercial, residential and industrial environments. But, most of the strategies and techniques have heavy limitations in the form of computational resources, lack of proper data analysis of the measured trained data, dependence of the motion of the objects, inability to differentiate one object from other, and also there is a concern over speed of the movement and Illuminacy. Hence, there is a need to draft, apply and recognize new techniques of detection that tackle the existing limitations.

Objective

A model based on Scalable Object Detection using Deep Neural Networks to localize and track people/cars/potted plants and many others in the camera preview in real-time. This is implemented in an android application and used handy in a mobile phone or any other smart device.

Motivation and State of Art

Humans learn to recognize objects or humans by learning starting from their birth. Same idea has been utilized by incorporating the

intelligence by training into a camera using neural networks and TensorFlow. This enables to have the same intelligence in cameras, which can be used as an artificial eye and can be used in many areas such as surveillance, detection of objects/things etc.,

Literature Review

Currently, it is difficult to know when and where people occupy a building. The part of the difficulty arises due to the fact that the current sensor technology is not utilized to the full efficiency and also, due to the lack of utilization of proper data analysis methods. The comforting fact is that with the advent of 21st century, there has been a vast improvement in the sensor technology and arrival of the Internet of Thing (IoT) devices [1]. In an environment with clutter and noise, detection is even more challenging. One of the difficulties presented in literature for detection is to identify a human when stationary. This is a common problem for any sensor that is based on the reflection of acoustic, optical, or electromagnetic wave off a surface. However, the issue of stationary humans being identified as objects has been solved in my project. For example, fast moving objects in real time may cause confusion in identification or classification by computer vision techniques. At the tracking level, objects may be stationary for no apparent reasons or they may move in any direction spontaneously. This makes the tracking problem particularly challenging.

Furthermore, a particular type of technology may have difficulties meeting all necessary requirements in various lighting conditions, or rainy, foggy and inclement weather conditions, not to mention that most cameras or sensors have a limited field of view to monitor traffic in all directions. In addition, the clutter background and complex moving patterns of all objects on urban streets demand sophisticated and accurate real-time processing of sensor inputs to avoid false detection and recognition.

In order to overcome the aforementioned technical challenges, “You look only once” (Yolo) [2] detection system has been used not only to speed up the detection process, but also higher accuracy has been obtained. Yolo has not been implemented with android before and I have implemented this with android. One of the applications and

advantages is that the android mobile devices are easily available with everyone, and in future, this detection system can be applied for Sousveillance [3].

Methodology

This is an enhanced version of https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android

You can directly clone the repository shared https://github.com/azainab/androidTFDetect/ and add the files below(but you have to make sure you have all the dependencies mentioned in the following sections added to your system):

> go to the folder where the repository has been downloaded using terminal and then type 'git init' to activate the repository for git.
>
> .pb files in the assets folder (you can download the .pb files from https://drive.google.com/file/d/0B4dPZkgiYl-SWW1TNTBpNUtKbGs/view?usp=sharing)

Note: Select all the files in the android studio and then right click and select git>add. If you face the error 'Failed to finalize session : INSTALL_FAILED_INVALID_APK: Split lib_slice_0_apk was defined multiple times' just do build>clean project and run again.

Otherwise, you can follow the whole procedure below and follow the process:

Let's start with the building process for Android. The core of the TensorFlow is written in C++. In order to build for android, JNI (Java Native Interface) has to be used to call the C++ functions like loadModel, getPredictions, etc. A .so (shared object) file will be built, which is a C++ compiled file and a jar file which will consist of JAVA API that calls for the native C++ and then JAVA API will be called to get things done easily.

Files needed:

> the jar(Java API) and a .so(c++ compiled) file.

We must have the pre-trained model file and a label file for the classification.

Softwares, dependencies and packages needed:

Android SDK & NDK

Bazel—primary build system for tensorflow

Get c/c++ in eclipse—easy for android developers

cython $ pip install—no-use-wheel—no-cache-dir Cython h5py

cv2 $ conda install –c conda-forge opencv (Make sure before you start with the android project you have all of the below requirements fulfilled)

Python3, tensorflow 1.0, numpy, opencv3.

The installation of cv2 might be tricky on some versions of operating systems, this has been explained in subsequent sections.

a. Making .so file

$* touch workspace $ bazel build -c opt

You need gradle to import git projects in android. Install gradle, using homebrew in mac. You can first use git init and and git pull to get the repository (https://github.com/tensorflow/tensorflow.git [3]) in a folder and then import the android project.

Go to the project folder in terminal and

$ bazel build -c opt //tensorflow/contrib
/android:libtensorflow_inference.so
—crosstool_top=//external:android/crosstool
—host_crosstool_top=@bazel_tools//tools/cpp:toolchain
—cpu=armeabi-v7a

b. Making JAR file

The next step is to build a jar file which will be added to the android application of TF detect and this required bazel which has been installed in the dependencies section.

$ bazel build //tensorflow/contrib
/android:android_tensorflow_inference_java

All the steps above can also be referenced from [4]. By the end of this step the android application is ready to be installed into a mobile or any other smart device and has TF Classify, TF detect and TF stylize as three separate applications.

c. Implementing Yolo Detector (the crux)

The implementation above doesn't use TF Detect. The note on the implementation says:

Note: Currently, in this build mode, YUV -> RGB is done using a less efficient Java implementation, and object tracking is not available in the "TF Detect" activity. Setting the build system to 'cmake' currently only builds libtensorflow_demo.so, which provides fast YUV -> RGB conversion and object tracking, while still acquiring TensorFlow support via the downloaded AAR, so it may be a lightweight way to enable these features.

Trying to implement the above and adding Yolo detector will help the detections run faster.

The Yolo is used to help the detections happen in a faster way.

➢ Open the downloaded git tensorflow project as mentioned above and hover to the android section tensorflow>examples>android. There might be few ugrades or installations might be required when you open the project in android. Accept all settings as default, except that the NDK version has to be 14 only.

➢ After cloning, the android package in your system might face few issues for which the resolves have been given in my blog:

If you face the error "Error running app: default activity not found." I

have mentioned the solution in my blog
http://ameemazainab.blogspot.qa/

➢ As already discussed pre-trained model is needed for this implementation. “Inception5h“ package is available online for download [5] and it has to be added to assets folder in the android studio. This package contains the trained images that will be needed for the identification of the images.

➢ After making all the changes above, run the app in android studio.

➢ TF detect is very slow hence Yolo detector is added to TF detect.

Step by step procedure has been explained below to help the TF detect work.

For Yolo to work you will need:

➢ Open the android package: tensorflow>examples>android

➢ Make changes private static final boolean USE_YOLO = true;

➢ Dark net and Darkflow installed (http://pjreddie.com/darknet/ and https://github.com/thtrieu/darkflow)

➢ We will make changes in the darkflow package to customize it as per our requirement. (Most of the changes will be applied in the python files. In the process, there will be few packages necessary to be installed, and we have to install them all to avoid errors).

➢ Darkflow runs setup.py file to add all dependencies.

➢ For darkflow one needs cythonize from cython.build and python

Installing cv2:

```
Installation of cv2 will be required to run help.py python
file in the darkflow package.
```

Refer to (http://www.pyimagesearch.com/2016/10/24/ubuntu-16-04-how-to-install-opencv/) $ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip $ unzip opencv.zip $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip $ unzip opencv_contrib.zip

virtual env keeps the python and opencv installations clean. Hence we install it here

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/get-pip.py ~/.cache/pip
```

virtualenv and virtualenvwrapper

```
$ export WORKON_HOME=$HOME/.virtualenvs
$ source /Library/Frameworks/Python.framework/Versions
/2.7/bin/virtualenvwrapper.sh
$ echo -e "\n# virtualenv and virtualenvwrapper" >>
~/.bashrc
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >>
~/.bashrc
$ mkvirtualenv cv -p python3
$ workon cv
```

ensure that you are in the cv environment now and follow the next steps:

```
$ pip install numpy
$ cd /Users/zuni/Downloads/darkflow-master/opencv-3.1.0
$ mkdir build
$ cd build
$ brew install cmake
$ wget http://www.cmake.org/files/v2.8/cmake-2.8.12.2-
Darwin64-universal.tar.gz && tar xfz cmake-2.8.12.2-
Darwin64-universal.tar.gz
$ export PATH="`pwd`/cmake-2.8.12.2-Darwin64-universal/CMake
2.8-12.app/Contents/bin":$PATH:
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
```

```
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules
\
-D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python \
-D ENABLE_PRECOMPILED_HEADERS=OFF \
-D BUILD_EXAMPLES=ON ..
```

By now you will see successful installation of cmake which is necessary for cv installation

```
$ make -j4
$ make clean
$ make
```

One might face few errors, which would look like

Error 1 make[1]: [modules/videoio/CMakeFiles /opencv_videoio.dir/all]

Error 2 make[2]:

videoio.h has to be disabled to help this work. We can try the code below, it helps in disabling quicktime

```
$ cmake -DWITH_QUICKTIME=OFF -DWITH_GSTREAMER=OFF
-DWITH_FFMPEG=OFF -DCMAKE_C_COMPILER=/usr/bin/clang
-DCMAKE_CXX_COMPILER=/usr/bin/clang++
-DCMAKE_BUILD_TYPE=Release .. ; make -j4
```

-go the build folder

```
$ cd /Users/zuni/Downloads/darkflow-master/opencv-
3.1.0/build
$ make
```

-Now one actually installs opencv3 in mac $ sudo make install

-To make sure workon runs on any terminal we need change few things:

```
$export WORKON_HOME=~/.virtualenvs
$VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
$source /Library/Frameworks/Python.framework/Versions
/2.7/bin/virtualenvwrapper.sh
$ pip install opencv-python
$ python
$ import cv2
```

-If this runs successfully cv2 is installed, and one can go ahead with the next steps.

Refer (https://github.com/thtrieu/darkflow)

Installation:

```
$ python3 setup.py build_ext -inplace
```

-Have a look at its options $ ./flow –h -One needs to install python using brew so we can link it $ brew install python $ export PATH=/usr /local/bin:/usr/local/sbin:~/bin:$PATH $ brew link python -go ahead with the next steps $ ./flow—model cfg/v1/yolo-tiny.cfg—load bin/yolo-tiny.weights

Note: The website has mentioned the names of the files which are not in the package. Make sure to have the files in the right path before you run.

```
$ ./ flow --model cfg/yolo-new.cfg
```

Note: No file named yolo-new.cfg

```
$ ./flow --model cfg/tiny-yolo-voc.cfg --load bin/tiny-yolo-
voc.weights --savepb –verbalise
```

-If you face this error “TypeError: makedirs() got an unexpected keyword argument ‘exist_ok’” it is because of the version of python being used. The code is written in python3 and you are using python2 version.

Make the changes below: os.makedirs(os.path.dirname(name), exists_ok=True) replace to try: os.makedirs(os.path.dirname(name)) except OSError as e: if e.errno != errno.EEXIST: raise or remove the line ‘exists_ok=True’. Just make it os.makedirs(os.path.dirname(name))

#In detectorActivity.Java you need to replace a line:

#We generate tiny-yolo-voc.pb using the flow command below. $ // ./flow—model cfg/tiny-yolo-voc.cfg—load bin/tiny-yolo-voc.weights—savepb—verbalise=True

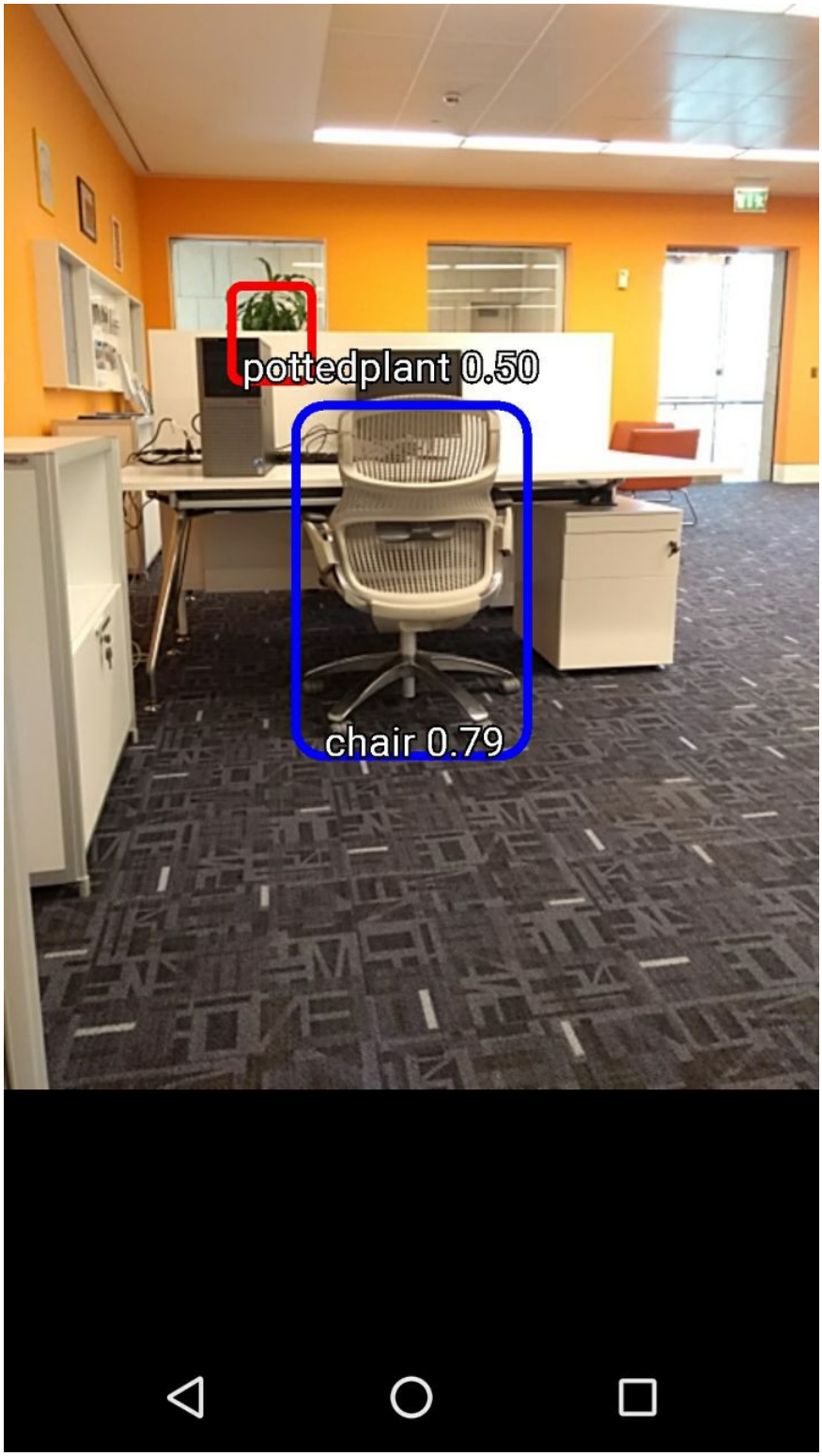
#Replace the line below: $ // private static final String YOLO_MODEL_FILE = “file:///android_asset/graph-tiny-yolo-voc.pb”; with this: $ private static final String YOLO_MODEL_FILE = “file:///android_asset/tiny-yolo-voc.pb”;

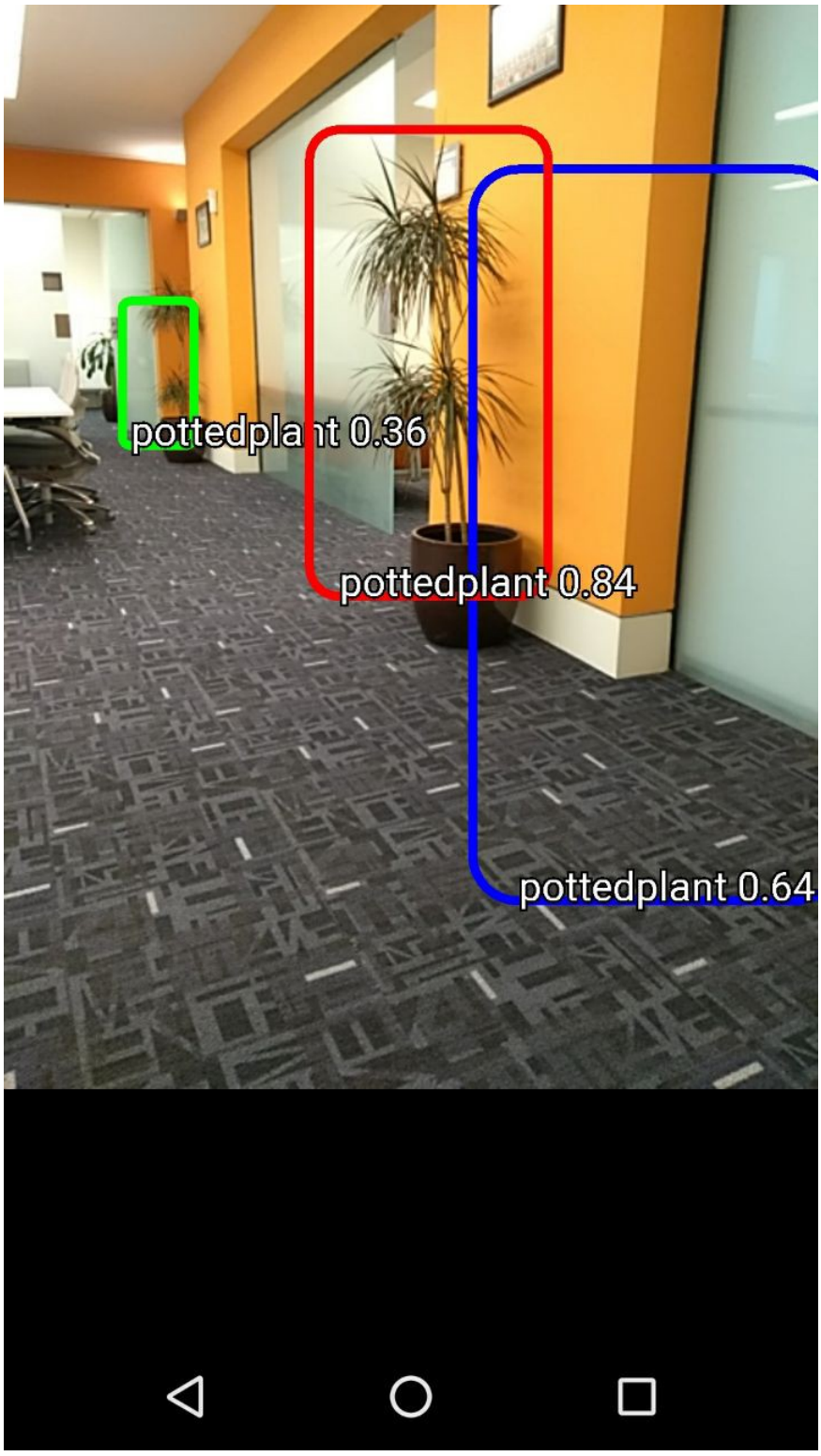
#The model is ready to be implemented

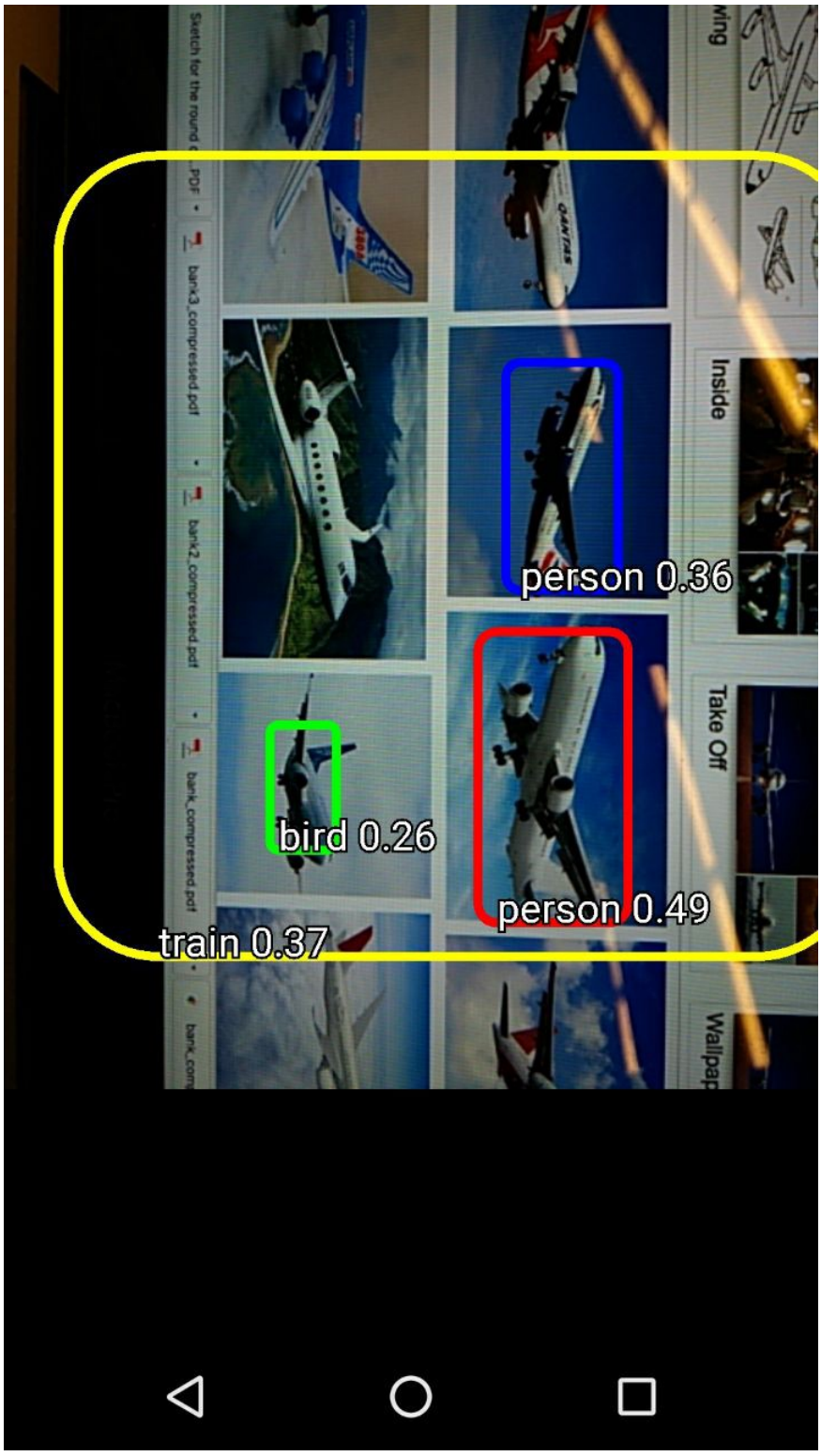
Results and conclusion

The following shown below are the screenshots captured from the android device. The 20 object categories were detected “aeroplane”,”bicycle”,”bird”,”boat”,”bottle”,”bus”,”car”,”cat”,”chair”,”cow”,”diningtable”,”dog”,”horse”,”motorbike”,”person”,”pottedplant”,”sheep”,”sofa”,”train” and “tvmonitor”. The accuracy is close to 100% for stationary object in the above mentioned categories in good

illumination. However when one of the factors such as light, speed of the object and object from different categories is changed the accuracy is reduced.




pottedplant 0.36
pottedplant 0.84
pottedplant 0.64



Note: The image above is taken from a computer screen which improves scope for research to improve the detection capabilities as you can see that an aeroplane is taken as a bird.