DETECTING HUMAN FACES AND EMOTIONS IN PHOTOS

DEVELOPED ON THE ANDROID PLATEFORM.

PROPOSED BY DR. SI CHEN - WEST CHESTER UNIVERSITY

AMAR BESSEDIK | CSC588 | 12/01/2016

TABLE OF CONTENT

MOTIVATION

Recognizing faces and emotions in people is a fascinating achievement of technology that burgeoned from combining artificial intelligence and psychology. It has many industry applications such as analyzing people faces in events, business in customer targeting and advertising and trying to understand autistic children among other things. This project helps explore two cloud-based services for mobile platforms which are Face API and Emotion API. It also explores mobile development technologies such as Android OS, Android Studio IDE, UI design, JAVA and XML languages.

NOTE:

1- Because of space constraint, only key pieces of Java code are included in this report. Full XML code as well as Full Java files can be seen on the in the unzipped project file.
2- Photos used in this report & app resource files serve only for test purposes in the academic environment.

APP NAME: FACEmotion

TEST PLATFORM: ANDROID LG K7 SMARTPHONE

INTRODUCTION

Microsoft Cognitive Services, previously based on what's known as Project Oxford, offer a set of cloud-based APIs for developers to make their applications more intelligent. Among these APIs are Computer Vision API, Speech Recognition API, Face API, Emotion API, Video API to name a few. The goal of this project is developing and implementing a smartphone application in order to be able to detect human emotions and faces in photos. To achieve the goal, two APIs will be needed: Emotion API and Face API respectively. In addition to that, the application will be developed on the Android platform using Android Studio IDE and the JAVA language in addition to XML.

In order to return results, the idea behind the application seems to be straightforward. The client (the actual smartphone application) makes pictures of human faces available with one of two methods. The first is by having the user browse their smartphone directories and select a photo, and the second is by having them take a picture of a human face using the smartphone camera. Once the photos are available, they will be sent to one of the above cited APIs through the Internet (A cloud-based service APIs). Computations will be performed on the photos and the results are sent back to the client for display on the screen. Figure 1 shows the general idea

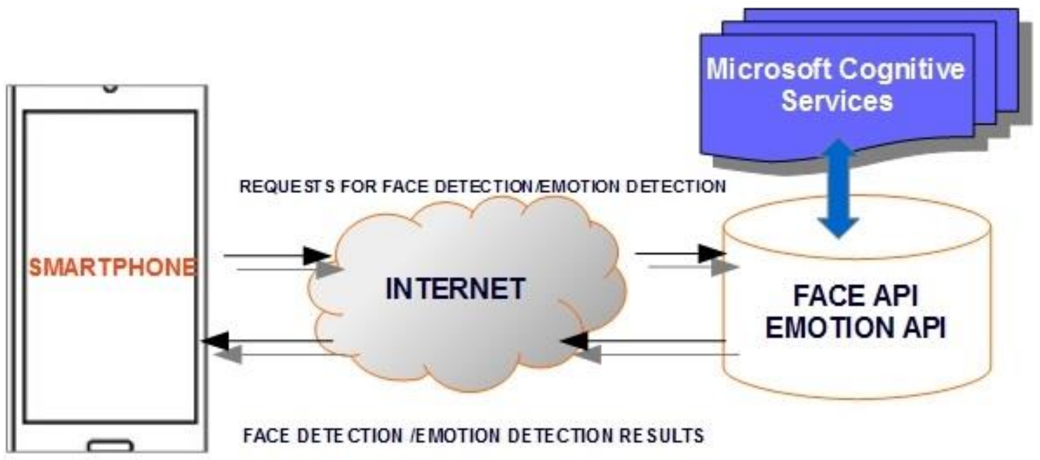

Microsoft Cognitive Services
REQUESTS FOR FACE DETECTION/EMOTION DETECTION
SMARTPHONE
INTERNET
FACE API
EMOTION API
FACE DETECTION /EMOTION DETECTION RESULTS

FIG1- GENERAL SCHEME

WORK ENVIRONMENT & DEFINITIONS

Face API: Used to detect human faces and compare similar ones, organize people into groups according to visual similarity, and identify previously tagged people in images.

Emotion API: Used to analyze human faces to detect a range of feelings.

Android: An operating system for smartphones developed by Google Inc. It powers hundreds of millions of devices around the world.

Android Studio: An IDE - Integrated Development Environment - for app development. Based on IntelliJ IDEA for code editing.

Android SDK: A software development kit that includes sample projects with source code, development tools, an emulator, and required libraries to build android applications.

Android Smartphone: A smartphone ran by the Android OS.

Java: In contrast to traditional computer applications that run on VMs (Virtual Machines), Java for android runs on Dalvik VMs or Android Runtime (ART). Also one noticeable difference is that in the traditional Java applications, the execution starts from a method called **main**(); whereas this method does not exist for the android apps, we use instead Activities (code representation for a single screen) and apps start from an activity called **MainActivity**.

XML: (Xtensible Markup Language) used to design the visual structure for a user interface, such as the UI for an activity or app widget. (1)

ARCHITECTURE & APP COMPONENTS

As specified in the above introduction, the application has two main tasks. The first one consists of **face detection** and the second of **emotions detection**. Before going to the detail of each operation, FIG2 shows the first façade of the app. The following scheme is a representation of how the app works as whole as well as showing its components. Detail about key components is provided in the next page.

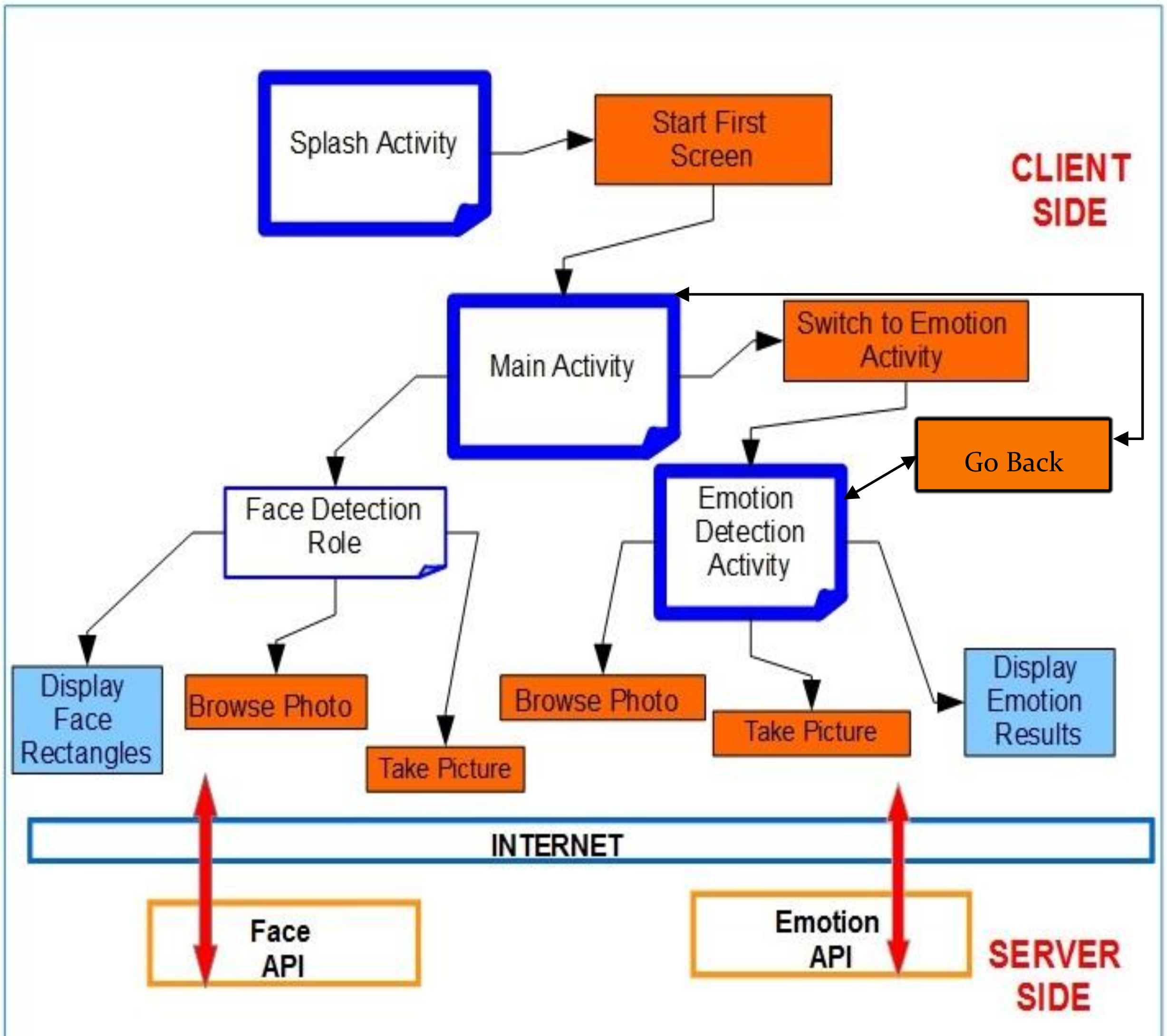
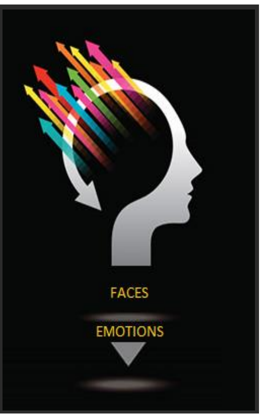


FIG2 – APP WORKINGS & COMPONENTS

DETAIL ON KEY COMPONENTS

When the user clicks on the app's icon, a splash screen is launched. With assistance of a thread the first screen sleeps for 2500ms then launches the main activity. This is intended to emulate the esthetic of app launching. The following code shows how that works and the splash screen image of the app is shown in the far right of the code.

```
Thread timer = new Thread(){
      public void run(){
         try{
             sleep(2500);
         }catch(InterruptedException e){
                e.printStackTrace();
         }finally{
          Intent intent = new Intent(
                      SplashActivity.this,MainActivity.class);
                      startActivity(intent);
            }
        }
    };
    timer.start();
}
```



After the main activity is launched, the user will be presented with 3 choices. Before enumerating the choices, it is worth noting that the main activity characterizes the screen that let the user ask for face detection after a photo is provided. As for the available choices at this stage, the user can **browse for a photo, take a picture** or go to the **emotion screen** (activity). The goals of browsing a photo and taking a picture is to provide data to the **Face API**. If the user choses to go to the emotion screen then, browsing and taking a picture are available choices in order to provide data to the **Emotion API** as well as the choice of going back to the face detection screen. FIG3 shows the UI design of face detection and emotion detection interfaces respectively.

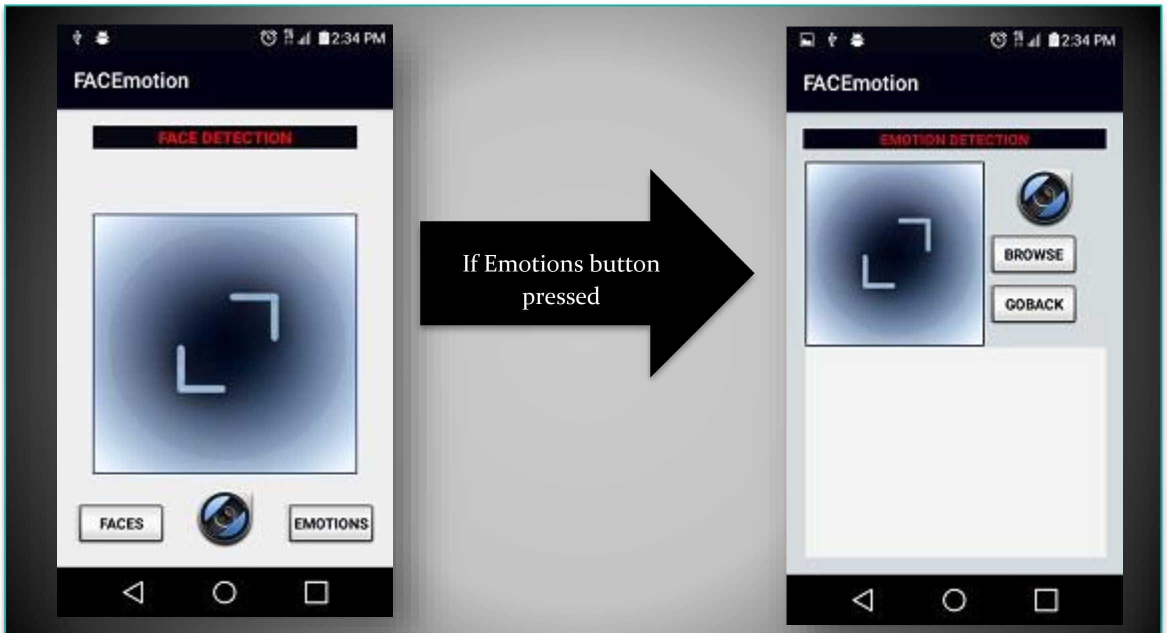


FIG3: USER INTERFACE DESIGN

FACE DETECTION

In order to detect a face, the user needs to provide a photo of a human face that needs to be vertically positioned. One of the powerful and useful aspects of Face API is its ability to detect multiple faces in a same photo. After detection, colored rectangles will be drawn around the faces and return the results to the client for display on the screen.

In order for this to happen, we need to provide a **SUBSCRIPTION "KEY"** to access the Face API and construct a – **FaceServiceClient** – object. This special object has a method called "**detect**" that returns the results as an array of faces "**Face**[]". The array of faces is returned with the help of the following piece of code:

```
com.microsoft.projectoxford.face.contract.Face[] result =
faceServiceClient.detect(
        params[0],
        true,         // return Face Id
        false,        // return Face Landmarks
        null           // return Face Attributes: a string like "age, gender"
);
```

As for the construction of the **FaceServiceClient object** by the following code**:**

```
private FaceServiceClient faceServiceClient = new
FaceServiceRestClient("3289657d188b4159b39df028555c5a19");
```

In order for the client to show detected faces, it uses a helper method that draws rectangles around the faces one by one as they are returned by the Face API. This method uses a bitmap copy of the target image and does not alter the original.

```
private static Bitmap drawFaceRectanglesOnBitmap(Bitmap originalBitmap, Face[]
faces) {
    Bitmap bitmap = originalBitmap.copy(Bitmap.Config.ARGB_8888, true);
    Canvas canvas = new Canvas(bitmap);
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setStyle(Paint.Style.STROKE);
    paint.setColor(Color.RED);
    int stokeWidth = 2;
    paint.setStrokeWidth(stokeWidth);
    if (faces != null) {
        for (Face face : faces) {
            FaceRectangle faceRectangle = face.faceRectangle;
            canvas.drawRect(
                    faceRectangle.left,
                    faceRectangle.top,
                    faceRectangle.left + faceRectangle.width,
                    faceRectangle.top + faceRectangle.height,
                    paint);
        }
    }
    return bitmap;
}
```

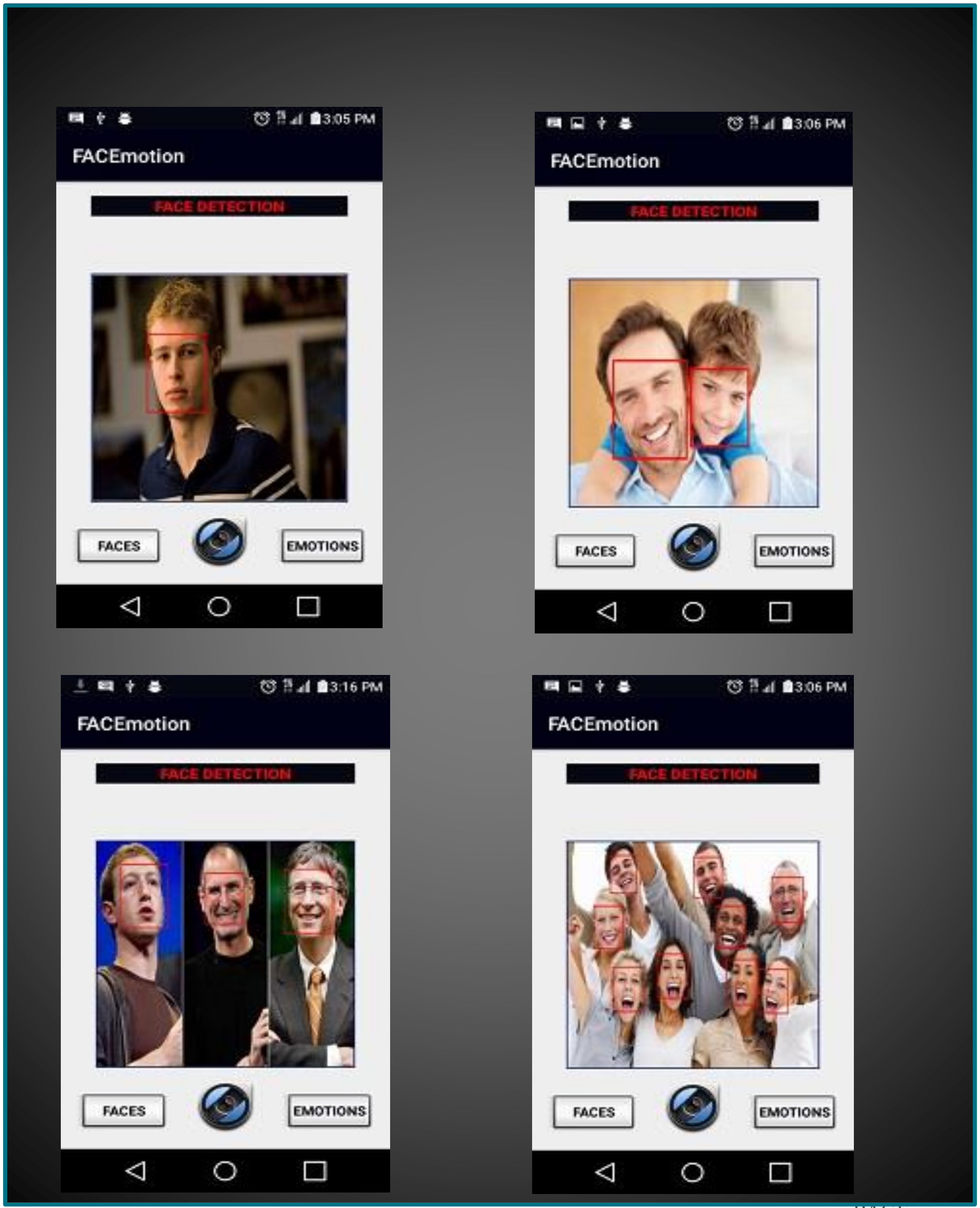
The helper method will be used by an asynchronous task called "**detectAndFrame**". The detail of this method is shown in the following code:

```
private void detectAndFrame(final Bitmap imageBitmap)
{
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
    ByteArrayInputStream inputStream =
          new ByteArrayInputStream(outputStream.toByteArray());
    AsyncTask<InputStream, String, Face[]> detectTask =
          new AsyncTask<InputStream, String, Face[]>() {
              @Override
              protected Face[] doInBackground(InputStream... params) {
                  try {
                        publishProgress("Detecting...");
      com.microsoft.projectoxford.face.contract.Face[] result = faceServiceClient.detect(
                    params[0],
                    true,         // return Face Id
                    false,        // return Face Landmarks
                    null           // return Face Attributes: a string like "age, gender"
                        );
                        if (result == null)
                        {
                            publishProgress("Detection Finished. Nothing detected");
                            return null;
                        }
                        publishProgress(String.format("Detection Finished. %d
                                        face(s) detected", result.length));
                        return result;
                    } catch (Exception e) {
                        publishProgress("Detection failed");
                        return null;
                    }
                }

                @Override
                protected void onPostExecute(Face[] result) {

                   detectionProgressDialog.dismiss();
                   if (result == null) return;
                   ImageView faceImageView = (ImageView)findViewById(R.id. faceImageView);
                   imageView.setImageBitmap(drawFaceRectanglesOnBitmap(bitmap, result));
                   bitmap.recycle();
                }
            };
    detectTask.execute(inputStream);
}
```

In order to display the results on the screen the following piece of code is used:

```
@Override
protected void onPostExecute(Face[] result) {

    if (result == null) return;
    ImageView imageView = (ImageView)findViewById(R.id.faceImageView);
    imageView.setImageBitmap(drawFaceRectanglesOnBitmap(bitmap, result));
    bitmap.recycle();
}
```

SAMPLES OF FACE DETECTION



EMOTION DETECTION:

In order to detect emotions in a face (including: Anger, Contempt, Fear, Disgust, Happiness, Sadness and Surprise.), the user needs to provide a photo of a human face; and the same principle as face detection, the photo needs to be vertically positioned. Emotion API is capable of detecting multiple emotions in multiple faces in a same photo. After detection, a list of emotions results is returned to the client for display on the screen as well as how many faces are detected. If no faces are detected, a message is displayed on the screen informing the user. In order for this to happen, we need to provide a **SUBSCRIPTION "KEY"** to access the Emotion API and construct an – **EmotionServiceClient** – object as follows.

```
private EmotionServiceClient emotionServiceClient = new
EmotionServiceRestClient("dad2de94aee4433d82c6b0a794e3c9c4");
```

The object has a method called "**recognizeImage**" that returns the results as a list of datatype **RecognizeResults** with each rectangle face containing the **scores attributes** as well as **positioning attributes**. The scores are as follows:

```
double[] SCORES = {face.scores.anger, face.scores.contempt,
                   face.scores.disgust, face.scores.fear,
                   face.scores.happiness, face.scores.neutral,
                   face.scores.sadness, face.scores.surprise};
```

And the positioning coordinates are:

```
int[] POSITIONS = {face.faceRectangle.left,
                   face.faceRectangle.top,
                   face.faceRectangle.width,
                   face.faceRectangle.height};
```

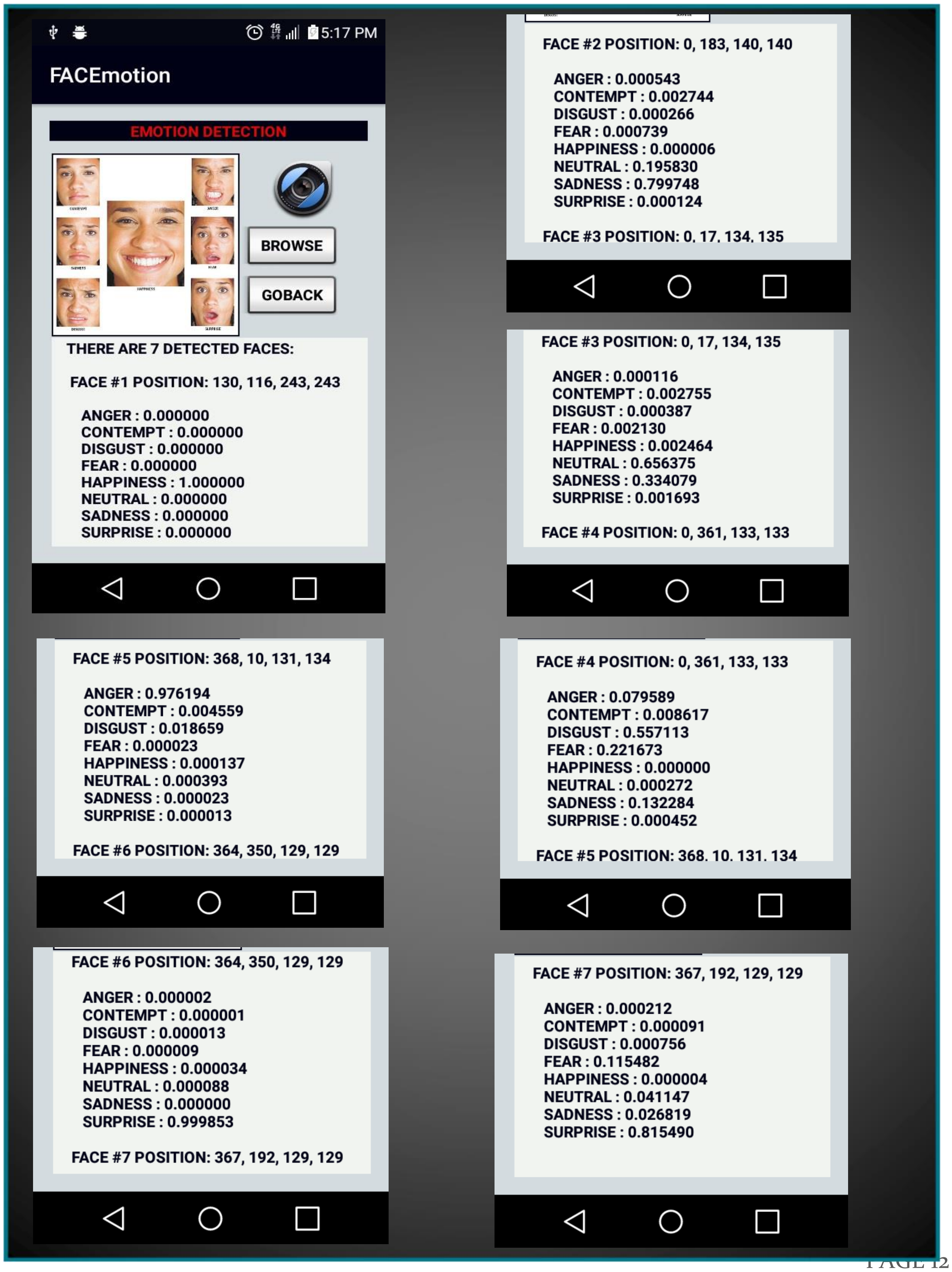
The list of results is returned using the following piece of code:

```
protected class DetectFacialEmotions extends AsyncTask<String, String,
List<RecognizeResult>> {
    @Override
    protected List<RecognizeResult> doInBackground(String... args) {
        try {
            // Put the image into an input stream for detection.
            ByteArrayOutputStream output = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, output);
            ByteArrayInputStream inputStream = new
ByteArrayInputStream(output.toByteArray());

            // Record emotions using the position of the faces in the image.
            result = emotionServiceClient.recognizeImage(inputStream);

            return result;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

As for the logic that displays the emotions scores of a scale [0, 1] and face positions in a photo. In case no faces detected, the app will display a message letting the user know. Important keywords are highlighted in red.

```
    @Override
    protected void onPostExecute(List<RecognizeResult> faces) {
        //For grammatical use: if 1 face use "IS" otherwise use "ARE"
        String verb = "ARE";
        //For grammatical use: If more than 1 face use the plural "S" otherwise
        //omit the "S"
        String singularOrPlural = "S";
        if(faces.size() == 1){
            verb = "IS";
            singularOrPlural = "";
        }

        int faceNumber = 0;
        if(faces.size() > 0) {
           String[] EMOTIONS = {"ANGER", "CONTEMPT", "DISGUST", "FEAR",
                                "HAPPINESS", "NEUTRAL", "SADNESS", "SURPRISE"};
            resultsTextView.append("\tTHERE "+verb+" "+faces.size()+" DETECTED
                                      FACE"+singularOrPlural+":\n\n");
            for (RecognizeResult face : faces) {

                double[] SCORES = {face.scores.anger, face.scores.contempt,
                                   face.scores.disgust, face.scores.fear,
                                   face.scores.happiness, face.scores.neutral,
                                   face.scores.sadness, face.scores.surprise};

                int[] POSITIONS = {face.faceRectangle.left,
                                   face.faceRectangle.top,
                                   face.faceRectangle.width,
                                   face.faceRectangle.height};

                //Increment the number of faces by 1 each time
                faceNumber++;
                resultsTextView.append(String.format(Locale.US,
                "\t FACE #" + faceNumber + " POSITION: %d, %d, %d, %d\n\n",
                 POSITIONS[0], POSITIONS[1], POSITIONS[2], POSITIONS[3]));

                for (int i = 0; i < EMOTIONS.length; i++) {

                    resultsTextView.append(String.format(Locale.US, "\t\t%s",
                                           EMOTIONS[i] + " : "));
                    resultsTextView.append(String.format(Locale.US, "%1$.6f\n",
                                           SCORES[i]));
                }
                resultsTextView.append("\n");
            }
        }else{
            resultsTextView.append("THERE ARE NO DETECTED FACES!");
        }
        emotionDetectionProgressDialog.dismiss();
    }

}//end DetectFacialEmotions class
```

A SAMPLE OF RESULTS ON EMOTIONS DETECTION



THERE ARE 7 DETECTED FACES:

FACE #1 POSITION: 130, 116, 243, 243

ANGER : 0.000000
CONTEMPT : 0.000000
DISGUST : 0.000000
FEAR : 0.000000
HAPPINESS : 1.000000
NEUTRAL : 0.000000
SADNESS : 0.000000
SURPRISE : 0.000000

FACE #2 POSITION: 0, 183, 140, 140

ANGER : 0.000543
CONTEMPT : 0.002744
DISGUST : 0.000266
FEAR : 0.000739
HAPPINESS : 0.000006
NEUTRAL : 0.195830
SADNESS : 0.799748
SURPRISE : 0.000124

FACE #3 POSITION: 0, 17, 134, 135

FACE #3 POSITION: 0, 17, 134, 135

ANGER : 0.000116
CONTEMPT : 0.002755
DISGUST : 0.000387
FEAR : 0.002130
HAPPINESS : 0.002464
NEUTRAL : 0.656375
SADNESS : 0.334079
SURPRISE : 0.001693

FACE #4 POSITION: 0, 361, 133, 133

FACE #5 POSITION: 368, 10, 131, 134

ANGER : 0.976194
CONTEMPT : 0.004559
DISGUST : 0.018659
FEAR : 0.000023
HAPPINESS : 0.000137
NEUTRAL : 0.000393
SADNESS : 0.000023
SURPRISE : 0.000013

FACE #6 POSITION: 364, 350, 129, 129

FACE #4 POSITION: 0, 361, 133, 133

ANGER : 0.079589
CONTEMPT : 0.008617
DISGUST : 0.557113
FEAR : 0.221673
HAPPINESS : 0.000000
NEUTRAL : 0.000272
SADNESS : 0.132284
SURPRISE : 0.000452

FACE #5 POSITION: 368, 10, 131, 134

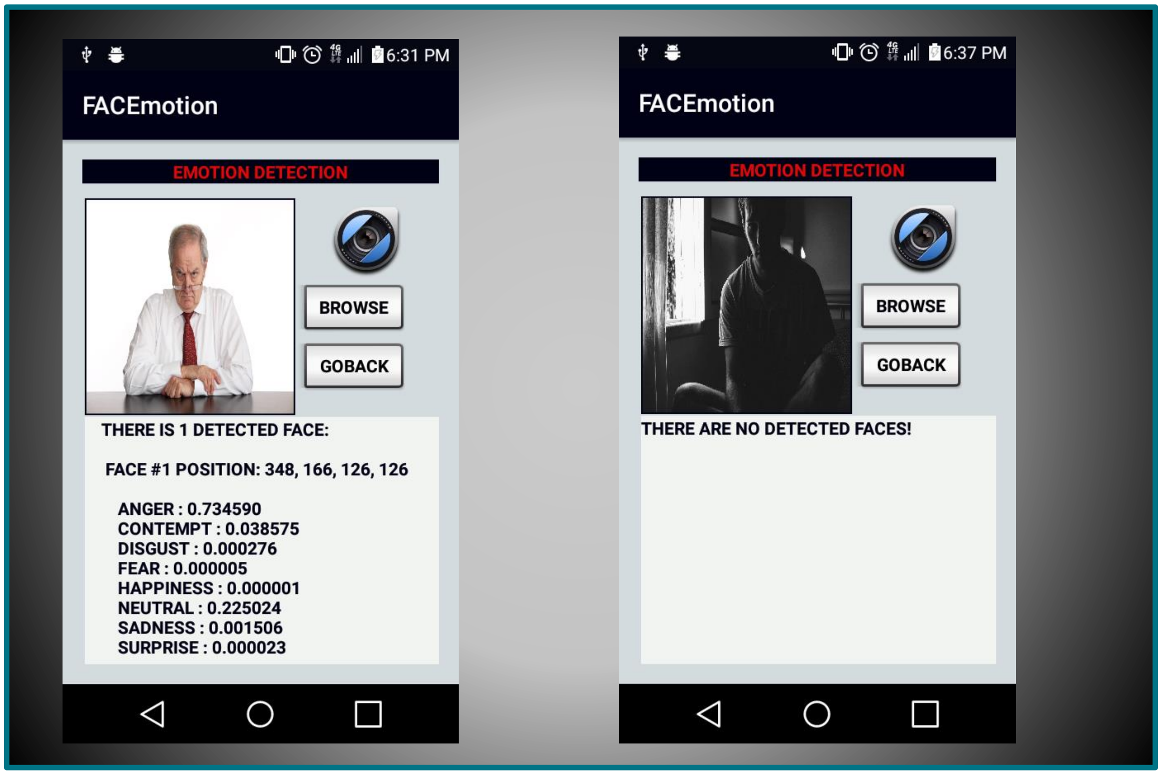
FACE #6 POSITION: 364, 350, 129, 129

ANGER : 0.000002
CONTEMPT : 0.000001
DISGUST : 0.000013
FEAR : 0.000009
HAPPINESS : 0.000034
NEUTRAL : 0.000088
SADNESS : 0.000000
SURPRISE : 0.999853

FACE #7 POSITION: 367, 192, 129, 129

FACE #7 POSITION: 367, 192, 129, 129

ANGER : 0.000212
CONTEMPT : 0.000091
DISGUST : 0.000756
FEAR : 0.115482
HAPPINESS : 0.000004
NEUTRAL : 0.041147
SADNESS : 0.026819
SURPRISE : 0.815490



THERE IS 1 DETECTED FACE:

FACE #1 POSITION: 348, 166, 126, 126

```
ANGER : 0.734590
CONTEMPT : 0.038575
DISGUST : 0.000276
FEAR : 0.000005
HAPPINESS : 0.000001
NEUTRAL : 0.225024
SADNESS : 0.001506
SURPRISE : 0.000023
```

THERE ARE NO DETECTED FACES!

CONCLUSION & FUTURE WORK

Working on this project has been enjoyable and motivating while discovering one of the advanced technologies. The app achieves its goals in detecting accurately all visible faces in a photo which is vertically positioned. It as well detects the seven emotions in each face and displays the results for the user in an ordered manner. This provides an insight on the increasing computing power of the mobile platforms. I find attractive the fact that reasonable using of this technology could help make more sense of our surroundings in social scenes, improve and optimize business, help doctors to better understand autistic children among many would be applications.

As for future work, I will definitely design and develop the logic behind face and emotions detections in videos. I am willing to try some of the other APIs such as Speech recognition API and Computer Vision API. This project motivates me to advance more my skills in app design and take this app one step further by refining its security and adding a database to keep track of results.

WEB REOURCES

https://www.asure.microsoft.com

https://www.microsoft.com/cognitive-services/en-us/apis

https://github.com/Microsoft/Cognitive-Emotion-Android

https://developer.android.com

https://source.android.com/devices/tech/dalvik/ (1)