1 Face Dectection API

Huawei, Samsung acnounced AI-featured inteligeng cellphones recently. However, we could release the AI-powered ability for older cellphone too.

2 Updated

1. android studio, 3.5.2
2. gradle, 5.1.1
3. external kotlin plugin, 1.3.51
4. JDK-8 build 211

3 Additional Note

Concerning to important role of the coming of Google's `AndroidX`, android.support (the old one), the libraries required in developing Adroid app have to be changed.

For instance,

```
com.android.support:appcompat-v7 ➡    androi
dx.appcompat:appcompat
com.android.support:design        ➡    com.go
ogle.android.material:material
```

Way to Introducing AndroidX

1. comment the implementation(s) in `build.grandle(Module app)`,
2. Choose [1. Library Dependcy] from

   ```
   [File] ➡ [Project Structure] ➡ [Dependenc
   ies](left menu) ➡ [+] (under right All de
   pendencies menu)
   ```

3. [step 1], input the library, to be added, (`com.google.android.material` and `androidx.appcompat`) and choose the artifact (`material` and `appcompat`) and the last versions.
4. [step 2], choose `Inplementation`, and press `[OK]` to proceed.
5.
6. After importing dependencies, add the following in `grandle.properties`:

   ```
   android.useAndroidX=true
   android.enableJetifier=true
   ```

or try to enable from `[refractor] (Main menu)→ [Migrate to AndroidX]`.

7. Now add the camera button in `activity_main.xml`:

```
<com.google.android.material.floatingacti
onbutton.FloatingActionButton
        android:id="@+id/revertCameraBu
tton"
        android:layout_width="wrap_cont
ent"
        android:layout_height="wrap_con
tent"
        android:layout_gravity="bottom|
end"
        android:layout_margin="16dp"
        android:src="@drawable/camera"
/>
```

In [43]:
```
import pandas as pd
from bayes_opt.util import Colours
```

In [2]:
```
df=pd.read_csv("androidx-class-mapping.csv")
df.head()
```

Out[2]:

	Support Library class	
0	android.arch.core.executor.AppToolkitTaskExecutor	androidx.arch.core.exe
1	android.arch.core.executor.ArchTaskExecutor	androidx.arch.core.
2	android.arch.core.executor.DefaultTaskExecutor	androidx.arch.core.ex
3	android.arch.core.executor.JunitTaskExecutorRule	androidx.arch.core.exec
4	android.arch.core.executor.TaskExecutor	androidx.arch.

In [22]:
```
def lib_query(old):
    new=df[df['Support Library class']==old]['A
    if len(new)>0:
        print('  Old: %s \n→ New: %s' %(old,new.
    else:
        print('No library found!')
```

In [25]:
```
lib_query('android.support.design.widget.Floati
```

```
  Old: android.support.design.widget.FloatingAction
Button
→ New: com.google.android.material.floatingactionbu
tton.FloatingActionButton
```

In [117]:
```
def lib_query_1(old):
    new=df[df['Support Library class'].str.cont
    if len(new)>0:
        for i in range(len(new)):
            #print('%s:  Old: %s \n  ➡ New: %s\
            print(Colours.BLUE,i,".  Old: ",new.
            print(Colours.RED,"   ➡ New: ",new.i
    else:
        print('No library found!')
```

In [118]:
```
lib_query_1('android.support.design.widget.Floa
```

```
 0 .  Old:  android.support.design.widget.FloatingA
ctionButton
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButton
 1 .  Old:  android.support.design.widget.FloatingA
ctionButtonImpl
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButtonImpl
 2 .  Old:  android.support.design.widget.FloatingA
ctionButtonImplLollipop
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButtonImplLollipop
```

In [120]:
```
lib_query_1('FloatingActionButton')
```

```
 0 .  Old:  android.support.design.widget.FloatingA
ctionButton
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButton
 1 .  Old:  android.support.design.widget.FloatingA
ctionButtonImpl
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButtonImpl
 2 .  Old:  android.support.design.widget.FloatingA
ctionButtonImplLollipop
    ➡ New:  com.google.android.material.floatingact
ionbutton.FloatingActionButtonImplLollipop
```

In [52]:
```
Colours.black
```

```
test
```

In [ ]:
```
```

4 Steps

Create new kotlin project, **app: facedetectapp, company: ai.kotlin.io**

0. If want to tyr Firebase feature, a mobile app development platform, enable certain plugins from right buttom menu,

[Configure] ➡ [Plugins] ➡ [Installed], all plugins beginning with Firebase.

1. Android target SDK-28 (Pie) required, install it from

```
[Tools] → [SDK Manager] → [Android SDK] →
[SDK platform] (☑ Show Packages Details)
        ▽ Android 9.0 Pie
         ☑ Android SDK Platform 28
```

   also install intel x86 Atom_64 System Image if want to test it on the fly.
2. create new project, App name: FaceDectectApp (Company name: ai.kotlin.io), as usual; and set the SDK conditions: **minSdkVersion 21, targetSdkVersion 28**.
3. modify **[build.gradle (Project: FaceDectectApp)]**:

```
buildscript {
    ...
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:g
radle:3.5.2'
        classpath "org.jetbrains.kotlin:kotl
in-gradle-plugin:$kotlin_version"
        classpath 'com.google.gms:google-ser
vices:4.3.3'
    }
}
allprojects {
    repositories {
        jcenter()
        maven { url "https://jitpack.io" }
        google()
    }
}
...
```

- modify **[build.gradle (Module: app)]**: here the entire list of dependent libraries:

```
...
dependencies {
    implementation fileTree(dir: 'libs', inc
lude: ['*.jar'])
    implementation "org.jetbrains.kotlin:kot
lin-stdlib-jdk7:$kotlin_version"

    //noinspection GradleCompatible
    implementation 'androidx.appcompat:appco
mpat:1.1.0'
    implementation 'com.google.android.mater
ial:material:1.2.0-alpha02'
    implementation 'androidx.test.espresso:e
spresso-core:3.2.0'
    implementation 'com.github.husaynhakeem:
android-face-detector:v1.2'
    implementation 'com.otaliastudios:camera
view:1.6.0'
    implementation 'com.google.firebase:fire
base-core:16.0.5'
    implementation 'com.google.android.gms:p
lay-services-vision:11.8.0'
    testImplementation 'junit:junit:4.12'
}

apply plugin: 'com.google.gms.google-service
s'
```

and **sync** project.

- Set up **[Firebase]** of the project:
  - Click **[Tools > Firebase]** to open the Assistant window.
  - Click to expand one of the listed features (for example, Analytics), then click the provided tutorial link (for example, Log an Analytics event).
    - Click the Connect to Firebase button] to connect to Firebase and add the necessary code to your app.
    - register Firebase
      - **[Firebase Console]** [events] [Android] and download the file, `google-services.json` , and put it on the directory, `$Project/app/` .
- To make Face detector app is easy by Firebase ML Kit's face detection API. Here, we introduce how to create such app based on "android-face-detector", a library on top of Firebase ML Kit's face detection API. In brief, face-detect app is designed in three steps:
  - Add a FaceBoundsOverlay on top of your camera view:

```
<FrameLayout  ...>
 <CameraView
     ... />
 <husaynhakeem.io.facedetector.FaceBou
ndsOverlay
     ... />
</FrameLayout>
```

- Define a FaceDetection instance and connect it to camera o device:

```
private val faceDetector: FaceDetector
by lazy {
 FaceDetector(facesBoundsOverlay)
}
...
cameraView.addFrameProcessor {
 faceDetector.process(Frame(
      data = it.data,
      rotation = it.rotation,
      size = Size(it.size.width, it.si
ze.height),
      format = it.format,
      isCameraFacingBack = cameraView.
facing))
}
```

- Setup firebase in this project.

5 Details

1 UI, acyivity_main.xml: use cameraview of otaliastudios:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.a
ndroid.com/apk/res/android"
             xmlns:tools="http://schemas.and
roid.com/tools"
             android:layout_width="match_par
ent"
             android:layout_height="match_pa
rent"
             tools:context=".MainActivity">

    <com.otaliastudios.cameraview.CameraView
            android:id="@+id/cameraView"
            android:layout_width="match_pare
nt"
            android:layout_height="match_par
ent"
            android:keepScreenOn="true" />

    <husaynhakeem.io.facedetector.FaceBounds
Overlay
            android:id="@+id/facesBoundsOver
lay"
            android:layout_width="match_pare
nt"
            android:layout_height="match_par
ent" />

    <com.google.android.material.floatingact
ionbutton.FloatingActionButton
            android:id="@+id/revertCameraBut
ton"
            android:layout_width="wrap_conte
nt"
            android:layout_height="wrap_cont
ent"
            android:layout_gravity="bottom|e
nd"
            android:layout_margin="16dp"
            android:src="@drawable/camera" /
>

</FrameLayout>
```

Within the last UI object, FloatingActionButton, we add a icon picture, called `camera.png` , which was placed within $Res/drawable sub-folder.

- kotlin part: init faceDectector with cmeraview, startup after

setup:

```
package io.kotlin.ai.facedetectapp

import android.support.v7.app.AppCompatActiv
ity
import android.os.Bundle
import com.otaliastudios.cameraview.Facing
import husaynhakeem.io.facedetector.FaceDete
ctor
import husaynhakeem.io.facedetector.models.F
rame
import husaynhakeem.io.facedetector.models.S
ize
import kotlinx.android.synthetic.main.activi
ty_main.*

class MainActivity : AppCompatActivity() {

    private val faceDetector: FaceDetector b
y lazy {
        FaceDetector(facesBoundsOverlay)
    }

    override fun onCreate(savedInstanceState
: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_mai
n)
        setupCamera()
    }

    private fun setupCamera() {
        cameraView.addFrameProcessor {
            faceDetector.process(Frame(
                data = it.data,
                rotation = it.rotation,
                size = Size(it.size.width, i
t.size.height),
                format = it.format,
                isCameraFacingBack = cameraV
iew.facing == Facing.BACK))
        }
        // Toggles the facing value between
Facing.FRONT and Facing.BACK.
        revertCameraButton.setOnClickListene
r {
            cameraView.toggleFacing()
        }
```

```
        }

        override fun onResume() {
            super.onResume()
            cameraView.start()
        }

        override fun onPause() {
            super.onPause()
            cameraView.stop()
        }

        override fun onDestroy() {
            super.onDestroy()
            cameraView.destroy()
        }
    }
```

Happy ending ...

In [ ]:
```
1
```