

姓名： 李嘉梁 班级： 计算机 211 学号： 211302104

题目： 专注于人脸计算的计算机视觉安卓应用（FaceCV）

开发环境：Android Studio Giraffe | 2022.3.1 Patch 1

compileSdk:33;Java version:17;  
camerax version:1.0.0-beta07;  
pytorch android lite:1.12.1;  
pytorch android torchvision lite:1.12.1;

## 一、 设计思路：

### 1. 任务列表：

- A. 实现点击按钮，存储照片，并提示存储成功
- B. 原有模型，点击按钮，计算并输出检测到的人脸的个数
- C. 新引模型，点击按钮，实现某一功能
- D. 改为 MVVM 结构

### 2. 任务排序：A->B->D->C

### 3. 实现结果：A/B/D/C 均已实现，额外实现任务 E.，任务列表补充：

- C. 点击按钮，实现功能切换(人脸检测<->表情推断)
- E. 点击按钮，实现摄像头转向

## 二、 设计原则：

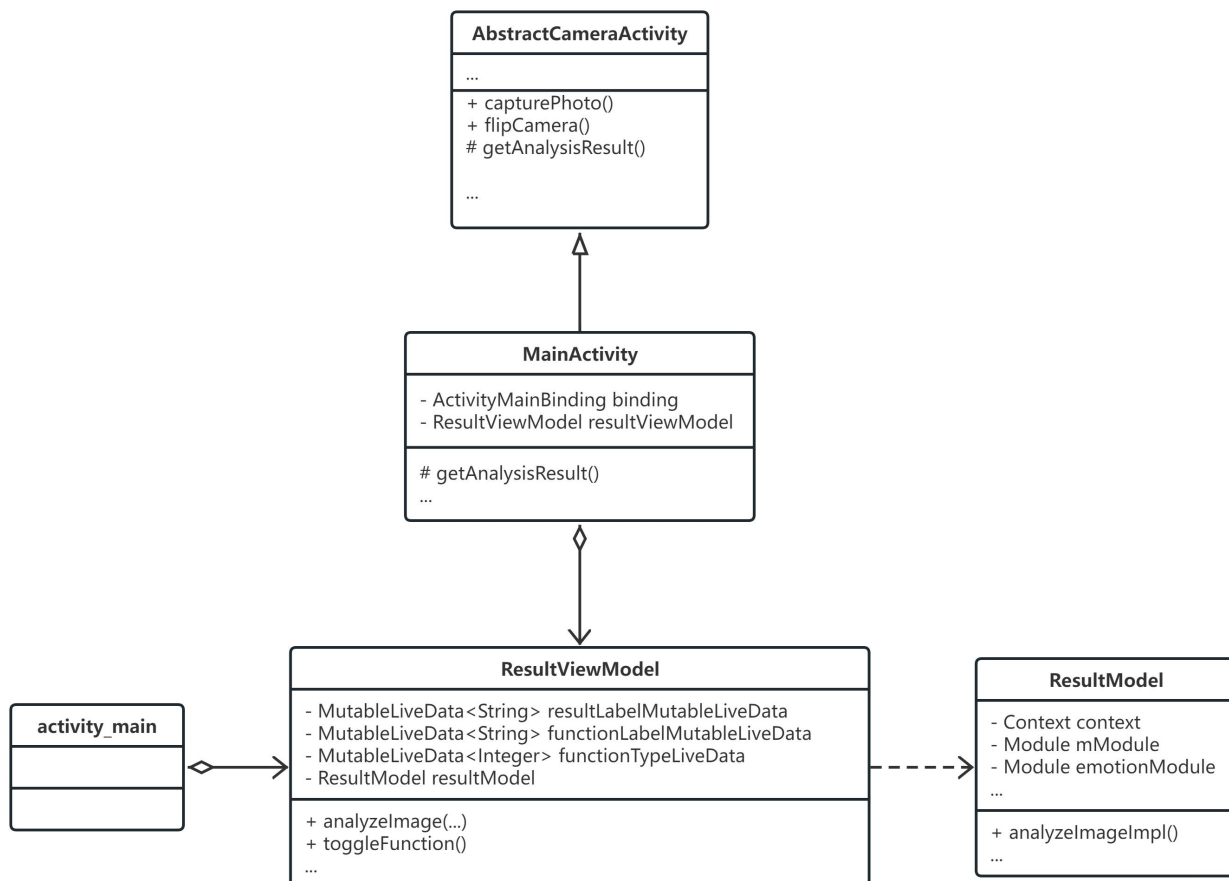
- 1. 数据为中心
- 2. 职能相分离

## 三、 设计思想：

- 1. 数据为中心：即以 VM 为数据对接中心；-->data-binding
- 2. 职能相分离：即使各个组件专注于属于自己的职能：-->MVVM
  - VM 专注于数据的业务逻辑；
  - M 专注于数据的处理逻辑；
  - V 专注于数据的接入和响应；
  - C: 专注于相机的基础/原始功能逻辑；
  - MA: 专注于 MVVM 的控制中枢逻辑，扩展相机高级功能逻辑；

## 四、 设计实现：

- 1. 逻辑结构：（下图显示最核心的组件、属性、方法）



下面介绍项目最核心的组件：

- 1) MainActivity.java: **MA**, 专注于 MVVM 的控制中枢逻辑：即负责承载 app 的核心控制架构（data-binding 绑定关系），精简后仅 50 行（带空格和注释）。扩展相机高级功能逻辑：即区别于原始相机的高级功能特性，更直观来说就是实现了可以调用模型的分析结果的功能。
  - 2) AbstractCameraActivity.java: **C**, 专注于相机的基础/原始功能逻辑：MA 的 prototype；实现了原始相机的基础功能。
  - 3) ResultViewModel.java: **VM**, 专注于数据的业务逻辑：使 MA 基本不必管辖数据的流动和功能的实现（MA 甚至不知道目前所处的功能模式是什么）。
  - 4) ResultModel.java: **M**, 专注于数据的处理逻辑：是将 app 高级功能特性实现的组件。
  - 5) activity\_main.xml: **V**, 专注于数据的接入和响应：是 app 数据流入和流出的端口。
3. 代码结构：

下面介绍项目最核心的控制流和数据流：

- 1) 推理模块控制流：

AbstractCameraActivity::getAnalysisResult() 抽象方法

-->调用

MA::getAnalysisResult() 调用 VM 获取结果

-->调用

VM::analyzeImage() 调用 M 获取结果

-->调用

M::analyzeImageImpl() 计算结果

2) 推理模块数据流:

**输入流:**

ACA::getAnalysisResult(拍照的用例, 旋转角, 朝向, 预览视图)

—>推送

MA::getAnalysisResult(...)

—>推送

VM::analyzeImage(...)+功能模式类型

—>推送

M::analyzeImageImpl(..., 功能模式类型)

**输出流:**

ACA::[推理结果]getAnalysisResult() 获取 boxing 部分显示在 V

<—返回

MA::[推理结果]getAnalysisResult()

<—返回

VM::[推理结果]analyzeImage() 获取计数/表情结果显示在 V (通过 resultLabelLiveData/结果标签 LD)

<—返回

M::[推理结果]analyzeImageImpl()

3) 功能切换模块数据流:

V::通过 a 显示可供切换的功能;通过 d 切换功能;

<—>双向绑定

VM:: a) functionLabelLiveData; 按键标签 LD

b) functionTypeLiveData; 功能模式 LD

c) getFunctionLiveData():

d) toggleFunction():

—>函数中推送形参

M::得知当前功能模式

4) 拍摄存储/翻转摄像头模块控制流:

V::通过 a, b 接口告知 C;

—>调用

C:: a) capturePhoto();

b) flipCamera();

4. 流程结构:

下面介绍项目推理模块的最核心的流程是:

1) 初始化相机配置和视图控件

2) 定时获取 image

3) 将 image 解码为 Bitmap

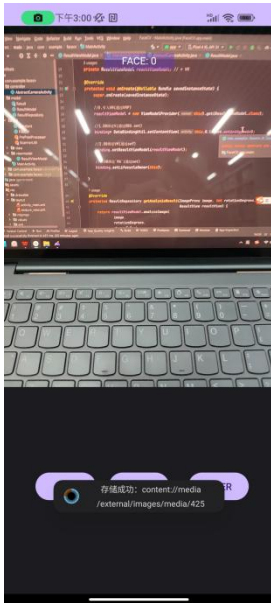
4) 用 TensorImageUtils 处理为模型输入张量

5) 加载 PyTorch 模型并进行推理

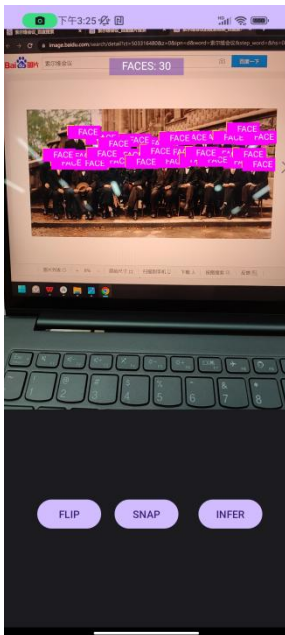
6) 处理和显示推理结果

## 五、 实现结果:

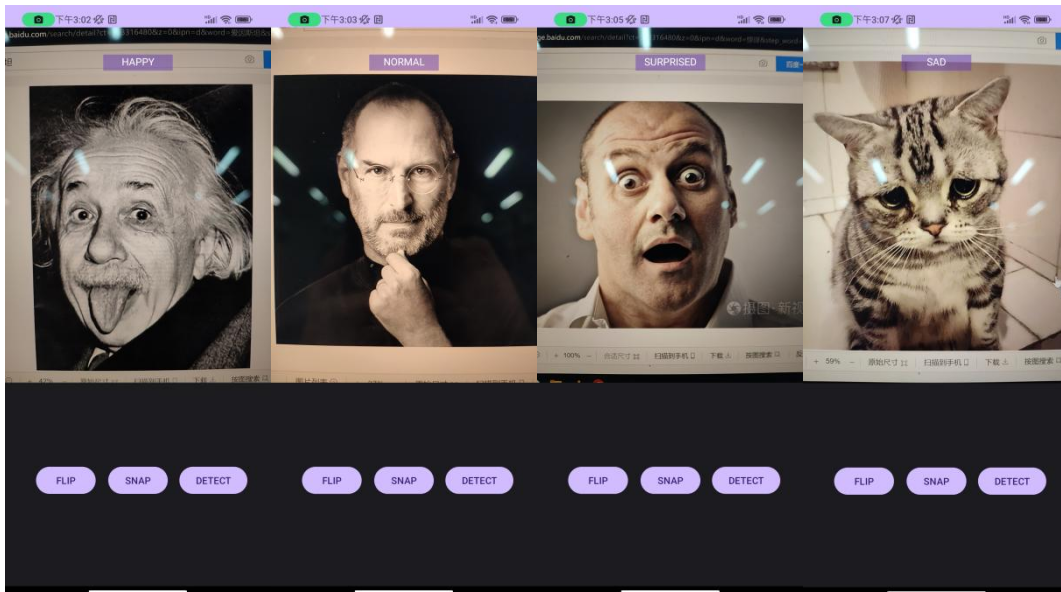
1. 拍摄存储功能: (点击“SNAP”, 屏幕下方提示存储成功并显示存储路径)



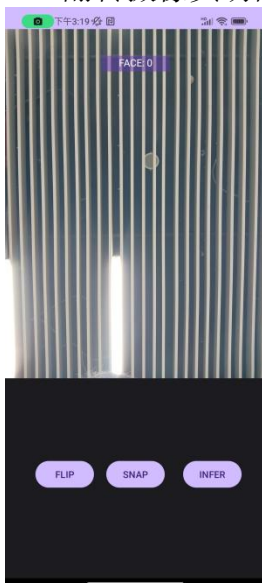
2. 人脸检测功能: (app 默认功能, 屏幕上方实时检测人脸数量)



3. 表情推断功能: (点击“INFER”，屏幕上方实时推断表情, 可点击“DETECT”切换回人脸检测功能)



#### 4. 翻转摄像头功能：（点击“FLIP”，实现摄像头翻转）



## 五、实现问题：

1. **问题：**想在运行过程中切换摄像头，但是无法切换，检查 Logcat 抛出异常：

java.lang.NullPointerException: Attempt to invoke virtual method 'android...

**解决：**经上网查询，推断问题出现在“FindViewById 必须在 onCreate() 内”，而我的切换摄像头实现需要重新调用 startCamera() [原本是在 onCreate() 中调用]，这个方法中又调用了 getCameraPreviewTextureView(), 而后者方法中又调用了 FindViewById()！但是现在我是在运行过程中调用，所以会出异常。因此解决思路是只允许 getCameraPreviewTextureView() 这个方法在第一次创建(onCreate())时调用，并将返回值保存在成员变量中，再次需要用只需访问成员变量即可。代码复现如下：

```
// PreviewView textureView = getCameraPreviewTextureView();
// * new : Only onCreate() that findViewById() is permitted
textureView = firstIn?getCameraPreviewTextureView():textureView;
firstIn = false;
```

2. **问题：**抛出异常：java.lang.IllegalStateException: Cannot invoke setValue on a background thread

**解决：**经上网查询，得知报错原因为非主线程中对 LiveData 数据进行 setValue() 更新数据，setValue() 必须在主线程中调用，不能在非主线程中调用，如果要在非主线程中调用，要使用 postValue() 接口，它可以在主线程中调用，也可以在子线程中调用。

3. **问题：**对接 controller 的按键无响应，抛出异常：

java.lang.IllegalStateException: Could not find method capturePhoto(View) in a parent or ancestor Context for android:onClick attribute defined on view class com.google.android.material.button.MaterialButton with id 'capture\_photo\_button'

java.lang.IllegalStateException: Could not find method flipCamera(View) in a parent or ancestor Context for android:onClick attribute defined on view class com.google.android.material.button.MaterialButton with id 'flip\_camera\_button'

**解决：**经上网查询，得知原因为 controller 的对应调用方法必须为 public，并且接收一个形参 View view。

4. **问题：**点击翻转摄像头无响应。

**解决：**推断相机用例配置与相机对象绑定在一起，翻转摄像头需要再次 startCamera() 重新绑定，经测试后假设得到验证。

5. **问题：**切换功能后，与当前功能显示输出不一致的问题，即会错位返回另一种功能显示输出 null 或数组越界。

**解决：**推断存在一个先后时间差  $\delta t = t_2 - t_1$ ，使得 VM 和 M 得到的功能指示不一致，具体来说，考虑如下场景，VM 已经在  $t_2$  切换到 Function2 了，但得到的结果还是 M 从  $t_1$  用 Function1 计算得到的输出。因此，存在至少两种思路：1. 设定临时功能指示 Instruction1，保证 VM 和 M 在同一个 Transaction 参照同一种功能指示。（成功，代码复现如下）2. 使用 synchronized 对功能加锁。（未成功）

```
// * 临时功能模式：保证 VM <-> M 功能一致性
Integer tempFunctionType = this.functionTypeLiveData.getValue();

ResultRepository result =
    resultModel.analyzeImageImpl(image, rotationDegrees, changeToBack,
        // * add
        resultView, tempFunctionType); // 调用分析图像方法

if(Objects.equals(tempFunctionType, FUNC_DETECT)){

    int numerationResult = result.getResults().size();
    String prefix = numerationResult <= 1 ? PREFIX_SINGULAR : PREFIX_PLURAL;
    this.resultLabelLiveData.postValue(prefix + numerationResult);

}

else if(Objects.equals(tempFunctionType, FUNC_INFER)){

    String emotionResult = result.getEmotionResult();
    this.resultLabelLiveData.postValue(emotionResult);

}
```

## 六、设计与实现总结：

通过本次实验我对 MVVM 的软件架构模式有了更深刻、更完整的理解，在 5 次代码重构中充分体会到了职能划分明确后代码结构的清晰明了、干净整洁、系统直观。从 MA 的 1000 行到 50 行，从一箩筐到体系化解离... 每个用思维重建的过程都相当于一次对 MVVM 知识点的复习、总结、呈现，使我完成了从原理理解到实践实现的知识迁移，正向使得原理与实现联系更加紧密，反向修补了理解中的知识漏洞。非常感激孙老师的生动引领教导！