



FaceAI 人脸识别 SDK

API & 说明文档

Email: FaceAISDK.Service@gmail.com



「简要说明」

FaceAI SDK包括人脸识别、活体检测、人脸录入检测以及人脸搜索，可快速集成实现Android端侧人脸识别，人脸搜索等功能。

SDK 支持Android[5,15] 所有功能都在设备终端离线执行，**SDK本身不用联网**，不上传不收集保存任何人脸信息敏感资料更具隐私安全

动作活体支持张嘴、微笑、眨眼、摇头、点头 随机两种组合验证（支持去除特定的动作），支持系统摄像头和各种UVC协议双目摄像头。**RGB静默活体**需配备宽动态值大于105Db成像清晰抗逆光摄像头。

下表统计1万人脸在不同设备的SDK初始化启动速度和一次人脸搜索速度：

设备型号	启动速度	搜索速度(毫秒)
小米 13	79 ms	66 ms
RK3568-SM5	686 ms	520 ms
华为 P8	798ms	678 ms
联想Pad2024	245 ms	197 ms

SDK不限制人脸库容量，但强烈建议分组以减少人脸搜索匹配误差。

3000测试人脸图链接: <https://pan.baidu.com/s/1RfzJlc-TMDbOlQMFKpA-tQ?pwd=Face> 提取码: Face

体验Android APK下载地址: <https://www.pgyer.com/faceVerify>

「集成方式」

目前SDK托管在Maven Central Repository,可通过gradle在线集成：

implementation 'io.github.faceaisdk:Android:VERSION.'

SDK集成演示Demo： https://github.com/FaceAISDK/FaceAISDK_Android

Demo默认gradle 版本 7.4.2 ， gradle插件版本 7.6.6 ， java17 , kotlin 1.9.22



「硬件配置要求」

FaceAISDK 对应CPU 配置要求不高，存储和内存大小请根据自身业务需要配置；关键在于摄像头成像能力，光线环境（同时注意摄像头保持物理干净，镜头不污染油污汗渍等）

一. 芯片，储存和内存要求

芯片只要是ARM架构的CPU都支持，4核2.0 GHZ的64位CPU足够，目前不支持X86等其他架构。

二. 摄像头参数要求

1. 低噪点和高感光能力，支持200万像素（1080P）预览画面，（默认使用640*480格式分析数据）

2. 宽动态范围（WDR）与背光补偿（BLC）结合

室内环境：≥95 dB，室内均匀充足光，不过曝不过暗

基础要求：≥105dB，可应对日常逆光场景（如门窗旁的逆光等）

严苛环境：≥120dB，适用于强逆光、户外直射环境（如露天环境等）

人脸搜索识别如果无法使用红外双目摄像头请使用补光感应灯补充脸部光线

3. 低照度性能,彩色模式：≤0.01Lux（F1.2），支持弱光环境清晰成像。

4. 帧率：30fps左右，确保动态人脸无拖影，不要于25fps.

三. 摄像头硬件等级Camera Hardware Level 说明

Hardware Level从宏观上描述Camera设备具备的能力等级，可以使用代码具体判断的，参考集成Demo 源码MyCameraXFragment。

LEVEL_FULL：完整手动控制，高帧率录制，性能强劲。推荐使用本等级以上

LEVEL_3: YUV重处理，多流联动，FULL的超集，最新旗舰机会配置本类型摄像头

摄像头管理源码CaneraX已经暴露在SDK集成Demo 中，你可以根据硬件平台特性自由修改完善，也可以完全自定义。



「API 分类简要说明」

API 我们简单分为人脸录入，人脸识别和人脸搜索，我们建议人脸图都通过SDK提取人脸特征向量更方便管理使用，Android 平台也提供人脸图片Bitmap 进行识别搜索（需通过FaceImageConfig配置本地存储路径）

人脸录入

BaseImageDispose：检测摄像头流中人脸角度合适且活体后返回裁剪好的人脸和活体分数

getBaseImageEmbedding：获取裁剪好的人脸特征向量编码，不保存人脸图到本地

saveBaseImageGetEmbedding：保存裁剪好的人脸到本地并返回人脸特征向量

FaceAIUtils.disposeBaseFaceImage：检测一张照片中是否含有人脸并返回裁剪出人脸部分和对应的人脸特征向量编码；用于非SDK BaseImageDispose录入的一张照片处理

人脸识别

FaceEmbedding.saveEmbedding：1:1 人脸识别保证人脸特征向量

FaceEmbedding.getEmbedding：1:1 人脸识别获取对应ID人脸特征向量

FaceProcessBuilder：人脸识别中各种参数设置Builder

FaceVerifyUtils.goVerifyWithImageProxy：通过视频预览流参数ImageProxy进行识别

FaceVerifyUtils.goVerifyWithBitmap：自定义管理摄像头视频帧转bitmap进行识别

FaceVerifyUtils.goVerifyWithNV21Bytes：自定义摄像头NV21 Byte[]流识别 **9.8上线**

FaceVerifyUtils.goVerifyWithIR：通过红外(可选)和RGB Bitmap进行人脸识别

FaceVerifyUtils.evaluateFaceSimi：两张静态人脸图中人脸的相似度(开放使用)

人脸搜索

SearchProcessBuilder：人脸搜索中各种参数设置Builder


FaceSearchEngine.runSearchWithImageProxy 通过摄像头预览流ImageProxy进行识别

FaceSearchEngine.runSearchWithBitmap 自定义管理摄像头视频帧转bitmap进行识别

FaceSearchEngine.runSearchWithIR：通过红外(可选)和RGB Bitmap进行人脸识别

FaceSearchEmbeddingManger：人脸搜索人脸向量特征库增删改查管理

FaceSearchImagesManger：人脸搜索人脸图片库增删改查管理

更多详细人脸搜索特征向量管理，人脸图管理参考下面API 



「人脸录入」

```
/**
 * BaseImageDispose 本类处理SDK通过摄像头预览流获自动化取角度正常的人脸图
 * context : 上下文context
 * mode: 人脸角度检测匹配模式
 * 2 精确模式 人脸要正对摄像头，严格要求
 * 1 快速模式 允许人脸方位可以有一定的偏移
 * 0 简单模式 允许人脸方位可以「较大」的偏移
 * BaseImageCallBack: 检测匹配成功回调 裁剪矫正好后的人脸图
 */
baseImageDispose(Context mContext, int mode, BaseImageCallBack callBack)
```

```
/**
 * 获取裁剪矫正好后的人脸对应的特征向量编码
 * @param bitmap 裁剪矫正好后的人脸Bitmap
 * @return 特征向量编码
 */
public float[] BaseImageDispose.getBaseImageEmbedding(Bitmap bitmap)
```

```
/**
 * 保存裁剪矫正好后的人脸底图到指定的本地路径，并返回人脸特征向量编码
 * @param bitmap 裁剪矫正好后的人脸底图
 * @param pathName 保存路径目录
 * @param fileName 保存的图片名称
 * @return 人脸特征向量编码
 */
public float[] BaseImageDispose.saveBaseImageGetEmbedding(bitmap, pathName, fileName)
```

```
/**
 * 检测一张照片中是否含有人脸并返回裁剪出人脸部分和对应的人脸特征向量编码; 用于非SDK
 * APi录入的一张照片处理
 *
 * @param mContext 上下文Context
 * @param bitmap 未裁剪矫正的人脸图bitmap
 * @param callBack 回调结果，正常success返回裁剪后的人脸bitmap和特征向量，否则返回错误信息
 */
fun disposeBaseFaceImage(mContext: Context, bitmap: Bitmap?, callBack: Callback?)
```

「人脸识别」

FaceProcessBuilder: 人脸识别中各种参数设置Builder，参数太多，详细见Demo
https://github.com/FaceAISDK/FaceAISDK_Android/blob/publish/faceAILib/src/main/java/com/faceAI/demo/SysCamera/verify/FaceVerificationActivity.java

```
/**
 * 1:1 人脸识别保存人脸特征向量编码到本地缓存文件
 *
 * @param context 上下文context
 * @param filename 本地缓存文件名称
 * @param embedding 人脸特征向量编码
 */
public static void saveEmbedding(Context context, String filename, float[] embedding)
```

```
/**
 * 从本地缓存文件中提取对应的人脸特征向量
 *
 *
 * @param context 上下文context
 * @param filename 本地缓存文件名称
 * @return 人脸特征向量编码
 */
public static float[] loadEmbedding(Context context, String filename)
```

```
/**
 * 通过视频预览流参数ImageProxy进行识别
 * @param imageProxy CameraX 的暴露的 ImageProxy
 * @param margin 暂时不用，预留参数
 */
public void goVerifyWithImageProxy(ImageProxy imageProxy, int margin)
```

```
/**
 * 通过图像数据帧转为bitmap流进行识别
 * @param rgbBitmap CameraX 的暴露的 ImageProxy
 */
public void goVerifyWithBitmap(Bitmap rgbBitmap)
```

```
/**
 * 分析两张静态人脸图中人脸的相似度
 * @param mContext 上下文context
 * @return 人脸相似度分数
 */
public float evaluateFaceSimi(Context mContext, Bitmap b1, Bitmap b2)
```



「人脸搜索」

SearchProcessBuilder:人脸搜索中各种参数设置Builder，参数太多详见

https://github.com/FaceAISDK/FaceAISDK_Android/blob/publish/faceAllib/src/main/java/com/faceAI/demo/SysCamera/search/FaceSearchMainActivity.java

```
/**
 * FaceSearchEngine 类中摄像头预览处理搜索
 *
 * @param imageProxy 实时采集的数据流 CameraX 的暴露的 ImageProxy
 * @param margin 预留参数
 */
fun runSearchWithImageProxy(imageProxy: ImageProxy, margin: Int)
```

```
/**
 * FaceSearchEngine类中摄像头预览处理搜索
 *
 * @param rgbBitmap 摄像头实时采集的数据流转为Bitmap，用于人脸搜索，活体检测（可选）
 */
fun runSearchWithBitmap(rgbBitmap: Bitmap)
```

```
/**
 * FaceSearchEngine类中摄像头预览处理搜索
 * @param irBitmap 实时IR摄像头采集的数据流转为Bitmap，用户IR活体检测
 * @param rgbBitmap 实时采集的数据流转为Bitmap，用于人脸搜索
 */
fun runSearchWithIR(irBitmap: Bitmap, rgbBitmap: Bitmap)
```

```
/**
 * FaceSearchEmbeddingManger
 * 保存一张裁剪矫正好的Bitmap到本地路径path,同时保存对应的人脸特征向量编码到人脸数据库
 * @param bitmap 裁剪并矫正好的Bitmap
 * @param path 路径目录
 * @param name 人脸FaceID String
 * @param faceEmbedding 人脸特征向量编码
 */
fun insertOrUpdateFace(bitmap: Bitmap, path: String, name: String, faceEmbedding: FloatArray)
```



```

/**
 * FaceSearchEmbeddingManger
 * 保存name(unique id)对应的人脸特征向量编码到人脸数据库
 * @param name 人脸FaceID String
 * @param faceEmbedding 人脸特征向量编码
 */
fun insertOrUpdateFaceEmbedding(name:String,faceEmbedding: FloatArray)

```

```

/**
 * 批量插入人脸特征向量编码到本地人脸数据库
 *
 * @param list 人脸特征向量编码List
 */
fun insertOrUpdateFaceEmbeddings(list:List<SearchEmbedding>)

```

```

/**
 * 删除某个人脸特征向量
 *
 * @param faceID 人脸FaceID String
 */
fun deleteFaceEmbedding(faceID:String)

```

```

/**
 * 删除所有人脸特征向量
 */
fun clearFaceEmbeddings()

```

```

/**
 * 查询某个人脸特征向量编码
 * @param faceID 人脸FaceID String
 * @return 人脸特征向量编码
 */
fun getFaceEmbedding(faceID:String):SearchEmbedding?

```

```

/**
 * 查询所有人脸特征向量编码
 *
 * @return 人脸特征向量编码List
 */
fun getFaceEmbeddings():List<SearchEmbedding>

```


FaceSearchImagesManger：人脸搜索人脸图片库增删改查管理(如果业务上本地不需要人脸图可以只管理人脸特征向量更快捷方便)

```
/**
 * 已经失效，移动到FaceSearchEmbeddingManger中
 * 保存一张裁剪矫正好的Bitmap到本地路径path,同时保存对应的人脸特征向量编码到人脸数据库
 */
fun insertOrUpdateFaceDep(bitmap: Bitmap,path: String, name:String,faceEmbedding: FloatArray)
```

```
/**
 * 插入单张没有处理过的人脸照片,异步检测裁剪矫正好后会通过callBack 返回对应裁剪好的人脸和特征向量 或失败提示
 *
 * @param bitmap 单张没有处理过的人脸照片
 * @param pathName 本地保存的路径名称
 * @param callBack 回调
 */
fun insertOrUpdateFaceImage(bitmap: Bitmap, pathName: String, callBack: Callback?)
```

```
/**
 * 删除path目录下的「所有」本地人脸图，同时删除数据库中所有的特征向量！！
 *
 * @param path
 * @return
 */
fun clearFaceImages(path: String): Boolean
```

```
/**
 * 删除path路径下的某张本地人脸图，同时删除数据库中对应的特征向量
 *
 * @param pathName
 * @return
 */
fun deleteFaceImage(pathName: String): Boolean
```

```
/**
 * 匹配到的大于 Threshold的所有结果，如有多个很相似的人场景允许的话可以弹框让用户选择
 * SearchProcessBuilder setCallBackAllMatch(true) 才有数据返回 否则默认是空
 */
@Override
public void onFaceMatched(List<FaceSearchResult> matchedResults, Bitmap searchBitmap) {
    //已经按照降序排列，可以弹出一个列表框
    Log.d("onFaceMatched",matchedResults.toString());
}
```

「常见问题说明」

1.如何提升 1:N人脸识别的准确率?

参考专题文章 <https://mp.weixin.qq.com/s/G2dvFQraw-TAzDRFlgdobA>, SDK不可用于金融, 高安全级别场景, 某些场景我们提供以下方法供业务侧避免搜索匹配错误无可选处理路径

```
/**
 * 匹配到的大于 Threshold的所有结果, 如有多个结果场景允许的话可弹框让用户选择
 * SearchProcessBuilder setCallBackAllMatch(true) 才有数据返回 否则默认是空
 */
@Override
public void onFaceMatched(List<FaceSearchResult> matchedResults, Bitmap
searchBitmap) {
    //已经按照相似度降序排列
    Log.d("onFaceMatched",matchedResults.toString());
}
```

2 .uniApp 原生插件支持

参考UTS demo项目集成 https://github.com/FaceAISDK/FaceAISDK_uniapp_UTS

目前已经支持iOS, Android 1:1人脸识别和人脸录入等, 细节可以修改原生部分代码重新打包实现。

3.人脸识别的阈值设置说明

人脸识别和搜索都有相应的API设置阈值setThreshold(0.88f) //阈值范围限 [0.75 , 0.95]
阈值设置越大结果可信度越高, 比如你设置为0.75可能和你同性别弟弟妹妹就能通过你的人脸识别但是越不是越高越好, 设置越高需要你录入的人脸品质高以及摄像头成像能力也高 (宽动态>105 Db可抗逆光)

4.FaceAI SDK 版权说明

FaceAI SDK 使用开源+自研封装实现, 非虹软(试用每年还要激活), Face++, 商汤 商业方案二次包装

所有功能都可充分体验验证, 更多请咨询:  FaceAISDK.Service@gmail.com

