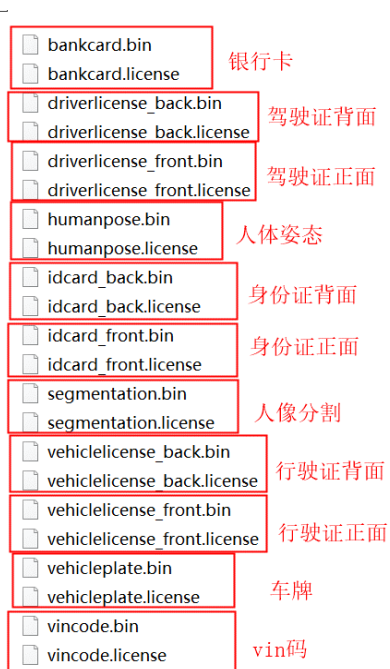


Android 端 openVisionSdk 集成说明

Android Studio 配置工程

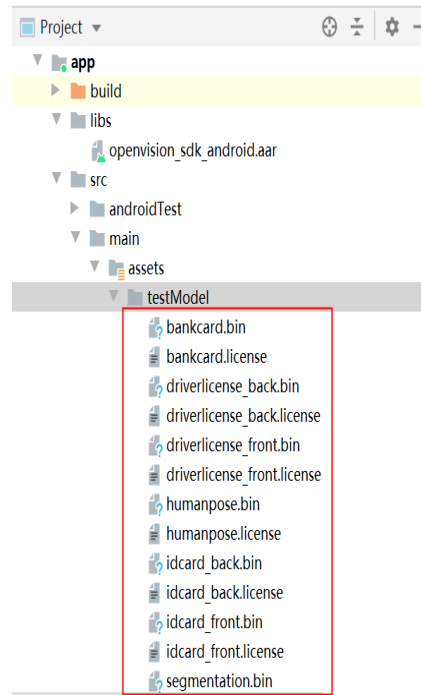
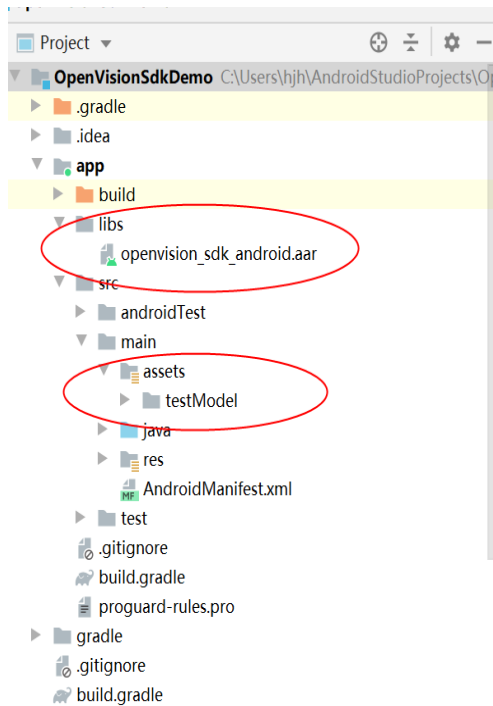
1、获取相关资源压缩包（由阿里云相关人员提供下载链接）后，解压压缩包，可看到

如下资源文件 **aar** 包及支持相关能力的 **bin**、**license** 文件。如下图：



2、把 **aar** 包拷贝到工程（此处截图以官方示例 **Demo** 为例子）的 **libs** 目录下，在工程

的 **assets** 目录下新建文件夹，把剩余的 **bin**、**license** 文件拷贝到该文件夹下，如下图：

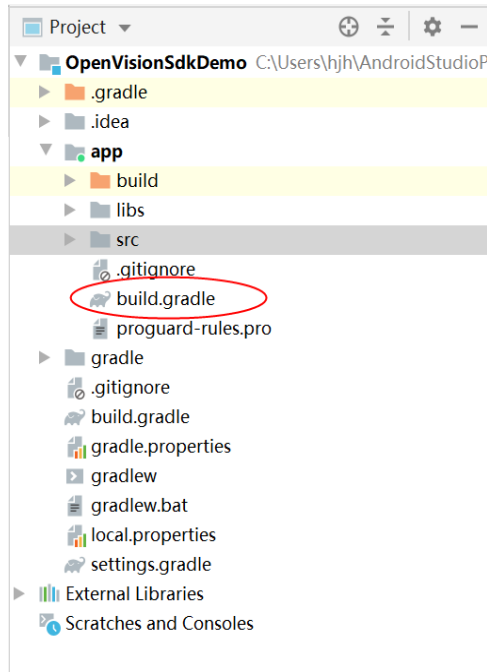


3、

(1)在配置文件 `AndroidManifest` 中增加相机和读写相关权限，`Android API23`（即 `Android 6.0`）及以上需要添加动态访问权限：

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

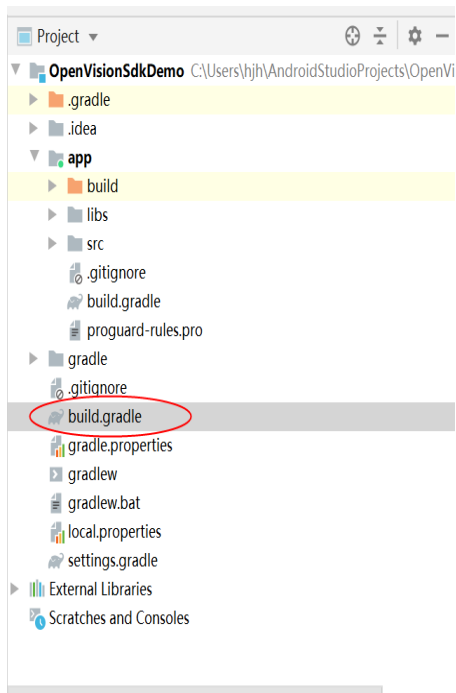
(2)在主工程的 `build.gradle` 文件相关配置设置及 `dependencies` 依赖，主工程的 `build.gradle` 文件在 `Project` 目录中位置如下图：



相关配置 `minSdkVersion` 最小为 18, `targetSdkVersion` 和 `compileSdkVersion` 可以按需配置:

```
android {
    defaultConfig {
        ndk {
            //设置支持的 SO 库架构
            abiFilters "armeabi-v7a", "arm64-v8a"
        }
    }
}
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    //相关 aar 包依赖
    implementation(name: 'openvision_sdk_android', ext: 'aar')
}
```

(3) 在 Project 的 `build.gradle` 文件中配置 `repositories`, 添加 `maven` 或 `jcenter` 仓库地址, `Android Studio` 默认会在 Project 的 `build.gradle` 为所有 `module` 自动添加 `jcenter` 的仓库地址, 如果已存在, 则不需要重复添加。Project 的 `build.gradle` 文件在 Project 目录中位置如图所示:



配置如下：

```
1. allprojects {
2.     repositories {
3.         google()
4.         jcenter() // 或者 mavenCentral()
5.
6.         flatDir {
7.             dirs 'libs'
8.         }
9.     }
10. }
```

4、相关功能实现，通过相机预览或相册选择图片实现银行卡、身份证（正、反面）、驾驶证（正、反面）、行驶证（正、反面）、车牌的信息识别。（注：相关功能的实现依赖于 aar 包是否包含该功能及项目中是否有对应的 bin 和 license 文件）。

声明相机类对象 CVCameraView、算法类对象 CVEngine、及算法结果回调 CVEngineCallback:

```
1. //声明相机类对象
2. private CVCameraView mCameraView;
3. //声明算法类对象
4. private CVEngine mCurEngine;
5. //算法结果回调
```

```

6. private CVEngineCallback mEngineCallback = new CVEngineCallback() {
7.     @Override
8.     public void onResult(CVEngine engine, CVResult result) {
9.         //result 对象可转换为 json 字符串, 按需获取需要数据, 需切回 UI 线程展示
10.        Log.i(TAG, engine.getClass().getSimpleName() + " result: " + JSON.toJSONString(result));
11.        //如果 result.error.code 等于 0 即为处理成功
12.        if (result.error.code == Error.ERROR_NONE && result.results != null) {
13.            //可获得算法处理过的图片以 Bitmap 形式返回
14.            if (result.image != null) {
15.
16.            }
17.        }
18.    };

```

实现相机内矩形框, 目的是为了把相关证件、卡片放矩形框内算法更准确识别, 可自己实现也可参考 Demo, 不添加识别框有可能会降低识别准确率。实例化相机 View, XML 文件中添加父布局加载相机 View 及矩形框:

```

1. //实例化相机类
2. float[] roi = new float[]{0.02f,0.218f,0.96f,0.294f};
3. mCameraView = new CVCameraView(this);
4. mCameraView.setRoi(roi);
5. //画类似银行卡、身份证的矩形框,目的算法可以更好识别信息, 可自己实现, 也可参考 Demo
6. ScanFrameView mScanRect = new ScanFrameView(this);
7. mScanRect.setRoi(roi);
8. //添加父布局加载相机 View 及矩形识别框
9. RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
10.    ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
11. RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
12. container.addView(mCameraView, lp);
13. container.addView(mScanRect, lp);

```

启动相机:

```

1. mCameraView.startPreview();

```

启动算法, 加载对应的算法库:

```

1. private void switchEngine(String engineName) {
2.     if (TextUtils.isEmpty(engineName)) {
3.         return;
4.     }
5.     if (mCurEngine != null) {
6.         mCurEngine.release();
7.         mCameraView.removeEngine(mCurEngine);
8.     }
9.     //正反面设置, 只有身份证、驾驶证、行驶证可设置

```

```
10.     boolean isFace = true;
11.
12.     CVConfig config = new CVConfig();
13.     switch (engineName) {
14.         case Engines.BANKCARD://银行卡
15.             config.license = MiscUtil.getAssetFileContent(this, "testModel/bankcard.license"); // getAssetFileContent 方法可参考 Demo 也可自己实现
16.             config.model = MiscUtil.getAssetFilePath(this, "testModel/bankcard.bin");
17.             mCurEngine = new Bankcard(this);
18.             break;
19.         case Engines.IDCARD://身份证
20.             if (isFace) { //正面
21.                 config.license = MiscUtil.getAssetFileContent(this, "testModel/idcard_front.license");
22.                 config.model = MiscUtil.getAssetFilePath(this, "testModel/idcard_front.bin");
23.             } else { //反面
24.                 config.license = MiscUtil.getAssetFileContent(this, "testModel/idcard_back.license");
25.                 config.model = MiscUtil.getAssetFilePath(this, "testModel/idcard_back.bin");
26.             }
27.             mCurEngine = new IDCard(this);
28.             break;
29.         case Engines.DRIVER_LICENSE://驾驶证
30.             if (isFace) { //正面
31.                 config.license = MiscUtil.getAssetFileContent(this, "testModel/driverlicense_front.license");
32.                 config.model = MiscUtil.getAssetFilePath(this, "testModel/driverlicense_front.bin");
33.             } else { //反面
34.                 config.license = MiscUtil.getAssetFileContent(this, "testModel/driverlicense_back.license");
35.                 config.model = MiscUtil.getAssetFilePath(this, "testModel/driverlicense_back.bin");
36.             }
37.             mCurEngine = new DriverLicense(this);
38.             break;
39.         case Engines.VEHICLE_LICENSE://行驶证
40.             if (isFace) { //正面
41.                 config.license = MiscUtil.getAssetFileContent(this, "testModel/vehiclelicense_front.license");
```

```

42.         config.model = MiscUtil.getAssetFilePath(this, "testModel
         /vehiclelicense_front.bin");
43.     } else { //反面
44.         config.license = MiscUtil.getAssetFileContent(this, "test
         Model/vehiclelicense_back.license");
45.         config.model = MiscUtil.getAssetFilePath(this, "testModel
         /vehiclelicense_back.bin");
46.     }
47.     mCurEngine = new VehicleLicense(this);
48.     break;
49.     case Engines.VEHICLE_PLATE://车牌
50.         config.license = MiscUtil.getAssetFileContent(this, "testMode
         l/vehicleplate.license");
51.         config.model = MiscUtil.getAssetFilePath(this, "testModel/veh
         icleplate.bin");
52.         mCurEngine = new VehiclePlate(this);
53.         break;
54.     case Engines.VIN://Vin 码识别
55.         config.license = MiscUtil.getAssetFileContent(this, "testMode
         l/vincode.license");
56.         config.model = MiscUtil.getAssetFilePath(this, "testModel/vin
         code.bin");
57.         mCurEngine = new VinCode(this);
58.         break;
59.     default:
60.         XLog.e(TAG, "unknown engine: " + engineName);
61.         return;
62.     }
63.     Error error = mCurEngine.init(config);
64.     //非 0 返回, 错误码及错误提示
65.     if (error.code != Error.ERROR_NONE) {
66.         final String prefix = mCurEngine.getClass().getSimpleName() + " E
         rror";
67.         JSON.toJSONString(error); //此处用的阿里 fastjson 库解析, 也可用其
         他解析, 得到错误码及错误信息。
68.     } else {
69.         mSelectedEngineName = engineName;
70.         mCameraView.addEngine(mCurEngine, mEngineCallback);
71.     }
72. }

```

在需要调用起算法时直接调用需要的功能点, 例如银行卡号识别:
switchEngine(Engines.BANKCARD);

Engines 类说明:

```
1. public class Engines {
2.     public static final String BANKCARD = "Bankcard";//银行卡
3.     public static final String IDCARD = "IDCard";//身份证
4.     public static final String DRIVER_LICENSE = "DriverLicense";//驾驶证
5.     public static final String VEHICLE_LICENSE = "VehicleLicense";//行驶证
6.     public static final String VEHICLE_PLATE = "VehiclePlate";//车牌
7.     public static final String VIN= "Vin";//车牌
8. }
```

拍照单张图片处理:

```
1. findViewById(R.id.take_picture).setOnClickListener(new View.OnClickListener() {
2.     @Override
3.     public void onClick(View v) {
4.         mCameraView.takePicture(mCurEngine, new CVEngineCallback() {
5.             @Override
6.             public void onResult(CVEngine engine, CVResult result) {
7.                 Log.d(TAG, engine.getClass().getSimpleName() + " takePicture "
8.                     + " result: " + JSON.toJSONString(result));
9.             }
10.        });
11.    }
12. });
```

Activity 各生命周期管理:

```
1. @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_preview);
       initView();
       switchEngine(Engines.BANKCARD);
   }
```

```
2. @Override
3.     protected void onResume() {
4.         super.onResume();
5.         mCameraView.startPreview();
6.     }
```

```
7. @Override
8.     protected void onPause() {
9.         super.onPause();
10.        mCameraView.stopPreview();
11.    }
```



```

1. @Override
2. protected void onDestroy() {
3.     super.onDestroy();
4.     if (mCurEngine != null) {
5.         mCurEngine.release();
6.         mCameraView.removeEngine(mCurEngine);
7.     }
8.     mCameraView.releaseCamera();
9. }

```

5、相关功能实现,人脸分割, 实现把人像抠出来的效果。

声明相机类对象 CVCameraView、算法类对象 SegmentationEngine:

```

1. //相机类
2. private CVCameraView mCameraView;
3. //算法处理类
4. private SegmentationEngine mSegmentation;

```

初始化 View, 在 XML 文件中添加父布局 (RelativeLayout、LinearLayout、ConstraintLayout 等均可), 加载相机 View 进父布局:

```

1. mCameraView = new CVCameraView(this);
2. RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
3.     ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
4. RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
5. container.addView(mCameraView, lp);

```

打开相机:

```

1. mCameraView.startPreview();

```

实例化算法, 并添加相机流结果回调:

```

1. CVConfig segmentationConfig = new CVConfig();
2. segmentationConfig.license = MiscUtil.getAssetFileContent(this, "testModel/segmentation.license");
3. segmentationConfig.model = MiscUtil.getAssetFilePath(this, "testModel/segmentation.bin");
4. mSegmentation = new SegmentationEngine(this);
5. Error error = mSegmentation.init(segmentationConfig);
6. if (error.code != Error.ERROR_NONE) {
7.     final String prefix = mSegmentation.getClass().getSimpleName() + " Error";
8.     String message = JSON.toJSONString(error) //此处用的阿里 fastjson 库解析, 也可用其他解析, 得到错误码及错误信息。
9.     return;
10. }
11. //算法结果回调
12. mCameraView.addEngine(mSegmentation, new CVEngineCallback() {

```

```

13. @Override
14. public void onResult(CVEngine engine, CVResult result) {
15.     Log.i(TAG, "Segmentation result: " + result);
16.
17.     SegmentationResult segmentationResult = (SegmentationResult) result;
18.     if (segmentationResult.error.code == Error.ERROR_NONE && segmentationResult.outputFrame != null) {
19.         //相机流获取的 bitmap 界面展示需在主线程操作
20.         if (segmentationResult != null && segmentationResult.outputFrame != null) {
21.             convertARGBFrameToBitmap(segmentationResult.outputFrame);
22.         }
23.     }
24. });

```

convertARGBFrameToBitmap 类:

```

1. private static Bitmap convertARGBFrameToBitmap(SegmentationResult.FrameData frame) {
2.     return Bitmap.createBitmap(frame.data, frame.width, frame.height, Bitmap.Config.ARGB_8888);
3. }

```

从相册选择图片进行处理:

```

1. private void chooseImage() {
2.     Intent getIntent = new Intent(Intent.ACTION_GET_CONTENT);
3.     getIntent.setType("image/*");
4.
5.     Intent pickIntent = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
6.     pickIntent.setType("image/*");
7.
8.     Intent chooserIntent = Intent.createChooser(getIntent, "Select Image");
9.     chooserIntent.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[] {pickIntent});
10.
11.     startActivityForResult(chooserIntent, PICK_IMAGE); // private static final int PICK_IMAGE = 1;
12. }

```

相册返回之后的回调:

```

1. @Override
2. protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
3.     super.onActivityResult(requestCode, resultCode, data);
4.     if (requestCode == PICK_IMAGE && resultCode == Activity.RESULT_OK) {
5.         if (data == null || data.getData() == null) {
6.             return;
7.         }

```

```

8.     try {
9.         InputStream is = getContentResolver().openInputStream(data.getData());
10.        Bitmap bitmap = BitmapFactory.decodeStream(is);
11.        if (bitmap != null) {
12.            SegmentationResult result = (SegmentationResult) mSegmentation.run(bitmap, null);
13.            if (result.error.code != Error.ERROR_NONE ) {
14.                final String prefix = mSegmentation.getClass().getSimpleName() + " Error";
15.                String message = JSON.toJSONString(result.error); //此处用的阿里 fastjson 库解析，也可用其他解析，得到错误码及错误信息。
16.            } else {
17.                SegmentationResult segmentationResult = (SegmentationResult) result;
18.                if (segmentationResult.error.code == Error.ERROR_NONE && segmentationResult.outputFrame != null) {
19.                    SegmentationResult.FrameData frameData = segmentationResult.outputFrame;
20.                    //选择相册图片处理之后的 bitmap
21.                    Bitmap bitmap1 = Bitmap.createBitmap(frameData.data, frameData.width, frameData.height, Bitmap.Config.ARGB_8888);
22.                }
23.            } }
24.        } catch (Exception e) {
25.            Log.e(TAG, "failed to open file", e);
26.        }
27.    }
28. }

```

Activity 各生命周期管理:

```

1.  @Override
2.  protected void onResume() {
3.      super.onResume();
4.      mCameraView.startPreview();
5.  }

```

```

1.  @Override
2.  protected void onPause() {
3.      super.onPause();
4.      mCameraView.stopPreview();
5.  }

```

```

1.  @Override
2.  protected void onDestroy() {
3.      super.onDestroy();
4.      if (mSegmentation != null) {
5.          mSegmentation.release();
6.      }

```

```

7.     mCameraView.removeEngine(mSegmentation);
8.     mCameraView.releaseCamera();
9.     }

```

6、相关功能实现,人体姿态, 获取人体描边的位置点。

声明相机类对象 CVCameraView、算法类对象 PoseDetectEngine:

```

1.     //相机类对象
2.     private CVCameraView mCameraView;
3.     //算法类对象
4.     private PoseDetectEngine mPoseDetect;

```

初始化 View, 在 XML 文件中添加父布局 (RelativeLayout、LinearLayout、ConstraintLayout 等均可), 加载相机 View 进父布局:

```

1.     mCameraView = new CVCameraView(this);
2.     RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
3.         ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
4.     RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
5.     container.addView(mCameraView, lp);

```

打开相机:

```

1.     mCameraView.startPreview();

```

切换相机前后摄像头:

```

1.     mCameraView.switchCamera();

```

实例化算法, 并添加相机流结果回调:

```

1.     CVConfig poseDetectConfig = new CVConfig();
2.     poseDetectConfig.license = MiscUtil.getAssetFileContent(this, "testModel/humanpose.license");
3.     poseDetectConfig.model = MiscUtil.getAssetFilePath(this, "testModel/humanpose.bin");
4.     mPoseDetect = new PoseDetectEngine(this);
5.     Error error = mPoseDetect.init(poseDetectConfig);
6.     if (error.code != Error.ERROR_NONE) {
7.         String prefix = mPoseDetect.getClass().getSimpleName() + " Error";
8.         //此处用的阿里 fastjson 库解析, 也可用其他解析, 得到错误码及错误信息。
9.         String Messages = JSON.toJSONString(error);
10.        return;
11.    }
12.
13.    mCameraView.addEngine(mPoseDetect, new CVEngineCallback() {
14.        @Override
15.        public void onResult(CVEngine engine, CVResult result) {

```

```

16. Log.i(TAG, "PoseDetect result: " + result);
17. Log.d(TAG, "showResult: " + JSON.toJSONString(result));
18. if (result != null && result.error.code == Error.ERROR_NONE) {
    PoseDetectResult mDetectResults;
    mDetectResults = ((PoseDetectResult) result);
19.     if (mDetectResults != null && mDetectResults.results != null) {
20.         for (Result[] results : mDetectResults.results) {
21.             for (Result result : results) {
22.                 for (PointF pt : result.pos) { //节点获取后需在主线程操作
23.                     //通过 for 循环获取所有节点信息，在画布上绘制（可参考 Demo 示例）及用于其他操作
24.
25.                 }
26.             }
27.         }
28.     } else {
29.         //未获取到结果
30.     }
31. }
32. }
33. });

```

Activity 各生命周期管理:

```

1. @Override
2. protected void onResume() {
3.     super.onResume();
4.     mCameraView.startPreview();
5. }

```

```

1. @Override
2. protected void onPause() {
3.     super.onPause();
4.     mCameraView.stopPreview();
5. }

```

```

1. @Override
2. protected void onDestroy() {
3.     super.onDestroy();
4.     if (mPoseDetect != null) {
5.         mPoseDetect.release();
6.     }
7.     mCameraView.removeEngine(mPoseDetect);
8.     mCameraView.releaseCamera();
9. }

```

7、其他：

文档中的 `JSON.toJSONString(result)` 依赖解析库（也可用 `gson` 解析代替，混淆参考其官方文档）：

1. implementation 'com.alibaba:fastjson:1.2.70'

混淆保持：

1. #aar 包混淆保持
2. -keep class com.aliyun.openvision.**{ *; }
3. -keep class com.alipay.xmedia.**{ *; }
4. -keep class com.ant.phone.**{ *; }
5. #fastjson 混淆相关，用 gson 参考 gson 官方文档混淆
6. -dontwarn com.alibaba.fastjson.**
7. -dontskipnonpubliclibraryclassmembers
8. -dontskipnonpubliclibraryclasses
9. -keep class com.alibaba.fastjson.**{ *; }
10. -keep class * implements java.io.Serializable { *; }
11. -keepattributes *Annotation
12. -keepattributes Signature

注：证件检测必须为所选证件项进行检测，包括正、反面的区别，否则会检测没反应或检测失败。