|  |
| --- |
|  |
| android 신분증 사진 특징 추출 라이브러리 v3.1.6.5\_a 사용 설명서 |
|  |

2023. 11. 20

한국인식산업

【변경이력】

|  |  |  |  |
| --- | --- | --- | --- |
| 변경일자 | 작성버전 | 변경  페이지 | 변경내용 |
| 2022.3.14 | ver 0.1 | 최초 작성 | 기전 2.2 버전에서 3.0 버전으로 인식율 개선  기존 native 라이브러리 형태에서 aar 으로 변경 |
| 2023.4.27 | Ver 0.2 |  | 3.1.5.4 설명  외국인증 및 보훈증 추가 |
| 2023.7.25 | Ver 0.3 |  | 3.1.6.4 설명  신분증 전체 사진에서 특징점 추출 추가 |
| 2023.8.24 | Ver 0.4 |  | 3.1.6.5 |
|  |  |  |  |
|  |  |  |  |

1. 배포 파일

프레임워크 : FaceprintH-3.1.6.5-release.aar

전체 크기 : 21M

세부내용 :

jni :

libFaceprintH.so

지원 아키텍쳐 : arm64-v8a , armeabi-v7a, x86\_64, x86

크기 : 9.1M

자바 클래스 :

classes.jar

asserts:

폴더 : kii\_Data

크기 : 12.1M

파일 :

Classifer ,

Mask.raw ,

MT ,

TFT ,

TFT\_SPADE ,

kii\_5\_landmarks.dat ,

seveneye\_kf3d.bin ,

seveneye\_kf3d.param ,

seveneye\_kf3r.bin , s

eveneye\_kf3r.param

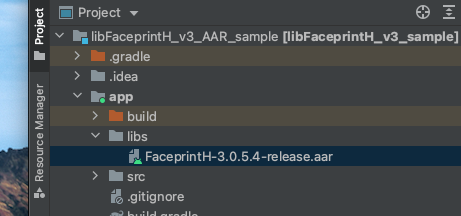
1. 적용 방법

* 기존 버전 2.2 업그레이드 할 때는 기존것 모두 지우고 aar 만 복사해야 합니다.
* 기존파일 삭제 :

jniLibs/libFaceprinte3x.so libc++\_shared.so ,

Data/Classifier,hologram\_detector.svm,kii\_5\_landmarks.dat, Mask.raw , MT , TFT , TFT\_SPADE , bio/face/Faceprint.java, FaceDetect.java

libs 폴더에 aar 파일 복사

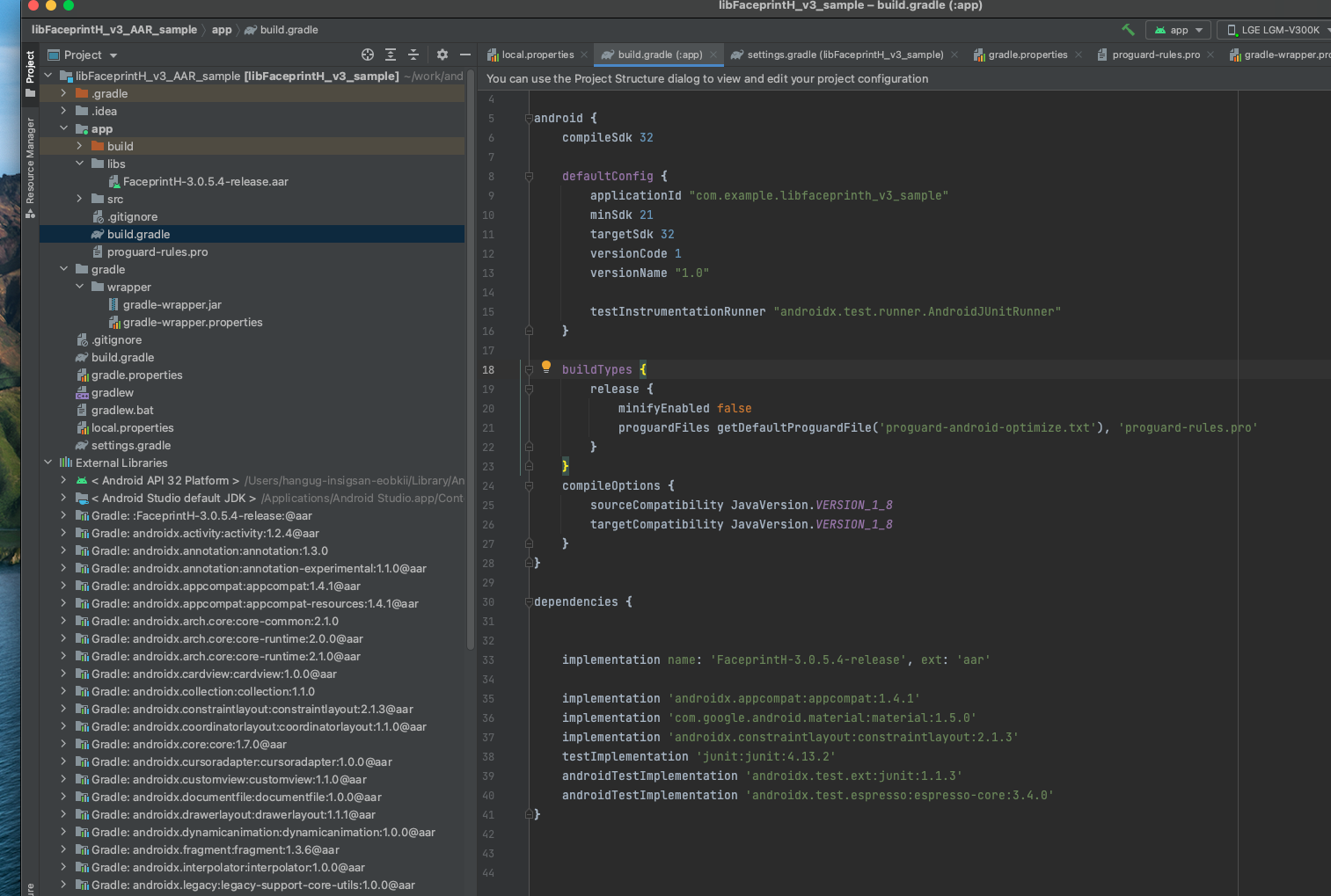


Build.gradle 에 아래 내용을 추가

dependencies {

implementation name: ‘FaceprintH-3.1.6.5-release’ , ext:’aar’

}



1. 라이브러리 함수 설명

* 클래스 함수 설명

1. 초기화 함수

**public int** FA\_Start(String path,Context context)

설명 : FaceDetect 클래스 생성시 반드시 해야 합니다.

app\src\main\assets\Data폴더의 파일을 메모리에 load 합니다.

파라메타 :

리턴 : 0 보다 크면 성공 작으면 실패

1. 종료함수

**public int** FA\_End()

설명 : FaceDetect 클래스 종료전에 반드시 해야 합니다.

메모리를 해제 합니다.

파라메타 :

리턴 : 0 보다 크면 성공 작으면 실패

1. 사진 품질 검사 함수

신분증 이미지에서 사진 부분을 크롭하여 bmp 또는 jpg 포멧으로 만들어 품질검사 함수를 호출한다.

BMP 포멧은 24bit 칼라 이미지

1. BMP 이미지 품질검사

**public** int FA\_QA\_BMP( byte[] buf , int nSize , int[] score\_flag)

설명 : 사진 품질검사.

파라메타 :

byte[] buf : BMP 이미지 버퍼 (신분증 이미지에서 사진 부분을 크롭하여 BMP 포멧으로 입력한다. )

int nSize : 이미지 버퍼의 크기

double[] score\_flag : 이미지 체크 스코어가 입력될 double[] 를 입력한다.

(**int** arr[] = **new int**[1]; )

리턴 : 0 : 인식가능

4 : 흑백

0 인경우 품질 양호 나머지는 품질 불량입니다.

1. JPG 이미지 품질검사

**public** int FA\_QA\_JPG( byte[] buf, int nSize , int[] score\_flag )

설명 : 사진 품질검사.

파라메타 :

byte[] buf : JPG 이미지 버퍼 (신분증 이미지에서 사진 부분을 크롭하여 JPG 포멧으로 입력한다. )

int nSize : 이미지 버퍼의 크기

double[] score\_flag : 이미지 체크 스코어가 입력될 double[] 를 입력한다.

( **int** arr[] = **new int**[1]; )

리턴 : 0 : 인식가능

4 : 흑백

0 인경우 품질 양호 나머지는 품질 불량입니다

1. 특징점 추출 함수
2. BMP 이미지 특징점 추출

**public int** FA\_Detect\_Ex\_bmp\_base64( byte[] buf, int nSize, byte []Feature , int cardType )

설명 : 특징점 추출

파라메타 :

byte[] buf : BMP 이미지 버퍼 (신분증 이미지에서 사진 부분을 크롭하거나 또는 신분증 전체 이미지를 BMP 포멧으로 입력한다. )

int nSize : 이미지 버퍼의 크기

byte []Feature : 특징점 추출될 버퍼 (크기는 반드시 4668 바이트로 설정한다 )

Int cardType : 신분증 종류를 입력한다 ( 주민증이면 1, 운전면허증은 2 )

리턴 : 음수 이면 오류

-10: 입력 데이타 오류

-401 : 이미지 포멧 오류

-501 : 눈동자 못 찿음

-3 : 이미지 오류

-5 : 얼굴 검출 오류

-1 : 기타 오류

0 : 특징점 추출 실패

1 이상이면 : 특징점 추출 성공 점수 ( 1점에서 100 점까지 있으면 숫자가 클수록 사진 품질이 좋다고 볼 수 있음 )

1. JPG 이미지 특징점 추출

**public int** FA\_Detect\_Ex\_jpg\_base64( byte[] buf, int nSize, byte []Feature , Int cardType )

설명 : 특징점 추출

파라메타 :

byte[] buf : JPG 이미지 버퍼 (신분증 이미지에서 사진 부분을 크롭하거나 또는 신분증 전체 이미지를 JPG 포멧으로 입력한다. )

int nSize : 이미지 버퍼의 크기

byte []Feature : 특징점 추출될 버퍼 (크기는 반드시 4668 바이트로 설정한다 )

Int cardType : 신분증 종류를 입력한다 ( 주민증이면 1, 운전면허증은 2, )

리턴 : 음수 이면 오류

-10: 입력 데이타 오류

-401 : 이미지 포멧 오류

-501 : 눈동자 못 찿음

-3 : 이미지 오류

-5 : 얼굴 검출 오류

-1 : 기타 오류

0 : 특징점 추출 실패

1 이상이면 : 특징점 추출 성공 점수 ( 1점에서 100 점까지 있으면 숫자가 클수록 사진 품질이 좋다고 볼 수 있음 )

1. 샘플 소스 작성

* 입력 이미지는 신분증 사진에서 ocr 라이브러리에서 신분증 사진 부분만 잘라낸 이미지를 사용한다.
* 작업 순서

1. 사진 품질 검사 실시 하여 리턴값이 0 인 경우만 다음으로 진행
2. 사진 특징점 추출 하여 리턴값이 1 에서 100 으로 나오는 경우 에만 정상 특징점 추출 된 경우 임
3. 정상 특징점 추출 되면 리턴값과 특징데이타를 서버로 전송
4. 샘플 이미지 준비 : id\_face.bmp , id\_face.jpg
5. 소스

package com.example.libfaceprinth\_v3\_sample;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;

import android.content.Context;

import android.os.Bundle;

import android.os.Environment;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import bio.face.FaceDetect;

public class MainActivity extends AppCompatActivity {

Activity act = this;

Context context = this;

FaceDetect fd;

int isFaceDetect = 0 ; // 초기화 유무

public void FaceDetectQA\_Start()

{

// 프로그램 시작시 한번 실행

//if( isFaceDetect == 1 ) return ;

fd = new FaceDetect(); // create

int rt = fd.FA\_Start("", context); // init

if( rt > 0) isFaceDetect = 1; // 초기화 성공

else isFaceDetect = 0; // 초기화 실패

Log.*i*("FA\_Demo", "FA\_Start " + String.*valueOf*(rt) );

if (rt == -1) {

Log.*i*("FA\_Demo", "License Time expire ");

}

if (rt == -2) {

Log.*i*("FA\_Demo", "License mac address fail ");

}

if (rt == 0) {

Log.*i*("FA\_Demo", "FA\_Start fail ");

}

Log.*i*("FA\_Demo", "License OK");

Log.*i*("FA\_Demo", "FA\_Start OK");

}

public void FaceDetectQA\_Stopt() {

// 프로그램 종료시 한번 실행

int rt = fd.FA\_End();

}

// FaceDetectBuf : 특징점 베이스 64 저장될 바이너리 버퍼 ( 4668 바이트 )

// cardType : 1:주민증 2:운전면허증

// 신분증 종류에 맞게 입력해야 각 행정 기관 매칭 서버에서 정상적으로 매칭 됨

public int FaceDetectQA( byte Buf[] ,int nFileSize , int iType, byte FaceDetectBuf[] , int cardType)

{

Log.*i*("FA\_Demo", "FA\_QA\_BMP start ");

int result = -1;

if( isFaceDetect == 0 )

return -1;

result = -20000;

Log.*i*("FA\_Demo", "FA\_QA\_BMP start 2 ");

// 품질검사 실시

int QA\_Result = 0 ;

int arr[] = new int[2];

if( iType == 1){

QA\_Result = fd.FA\_QA\_BMP(Buf, nFileSize,arr);

}

else {

QA\_Result = fd.FA\_QA\_JPG(Buf, nFileSize,arr);

}

Log.*i*("FA\_Demo", "FA\_QA\_BMP end ");

result = result - QA\_Result;

//QA 결과 출력

if(QA\_Result == 0)

Toast.*makeText*(act.getBaseContext(), "인식가능", Toast.*LENGTH\_LONG*).show();

else if(QA\_Result == 1)

Toast.*makeText*(act.getBaseContext(), "초점흐림", Toast.*LENGTH\_LONG*).show();

else if(QA\_Result == 2)

Toast.*makeText*(act.getBaseContext(), "반 사 광", Toast.*LENGTH\_LONG*).show();

else if(QA\_Result == 3)

Toast.*makeText*(act.getBaseContext(), "홀로그램" , Toast.*LENGTH\_LONG*).show();

else if(QA\_Result == 4) {

Toast.*makeText*(act.getBaseContext(), "흑백", Toast.*LENGTH\_LONG*).show();

}

else

Toast.*makeText*(act.getBaseContext(), "에러" + String.*valueOf*(QA\_Result) , Toast.*LENGTH\_LONG*).show();

if(QA\_Result==0 ) {// 품질검사통과하면

int detect\_rt = 0 ;

if( iType == 1){

detect\_rt = fd.FA\_Detect\_Ex\_bmp\_base64(Buf, nFileSize, FaceDetectBuf, cardType); //

}

else {

detect\_rt = fd.FA\_Detect\_Ex\_jpg\_base64(Buf, nFileSize, FaceDetectBuf, cardType); //

}

result = detect\_rt;

if( (detect\_rt >= 1) && (detect\_rt <= 100 ) )

{

Log.*i*("FA\_Demo", "특징점 추출 성공: 특징점 점수= " + String.*valueOf*(detect\_rt) );

}

else{

Log.*i*("FA\_Demo", "특징점 추출 실패: 에러= " + String.*valueOf*(detect\_rt) );

}

}

return result ;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

//return ;

FaceDetectQA\_Start();

findViewById(R.id.*btFaceDetectJpg*).setOnClickListener(

new Button.OnClickListener() {

public void onClick(View v) {

// System.out.println(System.getProperty("java.library.path"));

Log.*i*("FA\_Demo", "onClick");

FaceDetectQA\_Start(); // test

////////////////////////////////////////////////////////////////////

// 사진 이미지 파일 읽어오기

//////////////////////////////////////////////////////////////////////

InputStream myInput = null;

int iType = 2;

String path = "image/id\_face.jpg";

int nFileSize = 0;

int bufSize = 0;

byte Buf[] = null;

try {

InputStream is = context.getAssets().open(path);

bufSize = is.available();

Buf = new byte[bufSize];

nFileSize = is.read(Buf);

is.close();

Log.*i*(path, "read size = " + String.*valueOf*(nFileSize));

} catch (Exception ex) {

Log.*e*(path, ex.toString());

}

if (nFileSize < 1) {

Toast.*makeText*(act.getBaseContext(), "file read fail", Toast.*LENGTH\_LONG*).show();

return;

}

File file2 = Environment.*getExternalStorageDirectory*();

////////////////////////////////////////////////////////////////////

// 사진 품질검사 실시후 정상인 사진인 경우 특징점 추출 실시

//////////////////////////////////////////////////////////////////////

Log.*i*("FA\_Demo", "FaceDetectQA start");

byte FaceDetectBuf[] = new byte[4668]; // base 64

int cardType = 1 ; // 1:주민증 2:운전면허증

int result = FaceDetectQA( Buf, nFileSize , iType, FaceDetectBuf , cardType );

Log.*i*("FA\_Demo", "FaceDetectQA end "+ String.*valueOf*(result));

////////////////////////////////////////////////////////////////////

// 결과점수가 0보다 크면 FaceDetectBuf 의 값을 사용한다.

//////////////////////////////////////////////////////////////////////

if( (result >= 1 ) && (result <= 100) )

{

Toast.*makeText*(act.getBaseContext(), "인식 가능 검출 결과 " + String.*valueOf*(result), Toast.*LENGTH\_LONG*).show();

// 필요시 String 으로 변환

String bufStr = "";

String bufStr2 = "";

for( int i =0 ; i < 3000 ; i++)

{

bufStr = bufStr + Character.*toString*( (char) FaceDetectBuf[i] ) ;

}

for( int i =3000 ; i < 4668 ; i++)

{

bufStr2 = bufStr2 + Character.*toString*( (char) FaceDetectBuf[i] ) ;

}

Log.*i*("FA\_Demo", "특징점 점수="+String.*valueOf*(result));

Log.*i*("FA\_Demo", "특징점="+bufStr); // log 에 한번에 4668 바이트 출력이 안되어서 3000 + 1668 로 나누어서 출력 함

Log.*i*("FA\_Demo", "특징점="+bufStr2);

Toast.*makeText*(act.getBaseContext(), bufStr , Toast.*LENGTH\_LONG*).show();

}

else {

switch(result){

case -20001:

Toast.*makeText*(act.getBaseContext(), "사진품질:초점흐림", Toast.*LENGTH\_LONG*).show();

break;

case -20002:

Toast.*makeText*(act.getBaseContext(), "사진품질:반사광", Toast.*LENGTH\_LONG*).show();

break;

case -20003:

Toast.*makeText*(act.getBaseContext(), "사진품질:홀로그램", Toast.*LENGTH\_LONG*).show();

break;

case -20004:

Toast.*makeText*(act.getBaseContext(), "사진품질:흑백", Toast.*LENGTH\_LONG*).show();

break;

case -401:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:사진 포멧 에러", Toast.*LENGTH\_LONG*).show();

break;

case -201:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:메모리 확보 에러", Toast.*LENGTH\_LONG*).show();

break;

default:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:얼굴 검출 실패", Toast.*LENGTH\_LONG*).show();

break;

}

}

}

}

);

findViewById(R.id.*btFaceDetectBmp*).setOnClickListener(

new Button.OnClickListener() {

public void onClick(View v) {

// System.out.println(System.getProperty("java.library.path"));

Log.*i*("FA\_Demo", "onClick");

FaceDetectQA\_Start(); // test

////////////////////////////////////////////////////////////////////

// 사진 이미지 파일 읽어오기

//////////////////////////////////////////////////////////////////////

InputStream myInput = null;

String path = "image/id\_face.bmp";

int iType = 1; // 1: bmp, 2: jpg

int nFileSize = 0;

int bufSize = 0;

byte Buf[] = null;

try {

InputStream is = context.getAssets().open(path);

bufSize = is.available();

Buf = new byte[bufSize];

nFileSize = is.read(Buf);

is.close();

Log.*i*(path, "read size = " + String.*valueOf*(nFileSize));

} catch (Exception ex) {

Log.*e*(path, ex.toString());

}

if (nFileSize < 1) {

Toast.*makeText*(act.getBaseContext(), "file read fail", Toast.*LENGTH\_LONG*).show();

return;

}

File file2 = Environment.*getExternalStorageDirectory*();

////////////////////////////////////////////////////////////////////

// 사진 품질검사 실시후 정상인 사진인 경우 특징점 추출 실시

//////////////////////////////////////////////////////////////////////

byte FaceDetectBuf[] = new byte[4668]; // base 64

Log.*i*("FA\_Demo", "FaceDetectQA start");

int cardType = 2 ; // 1:주민증 2:운전면허증

int result = FaceDetectQA( Buf, nFileSize , iType, FaceDetectBuf , cardType );

// Log.i("FA\_Demo", "FaceDetectQA end");

Log.*i*("FA\_Demo", "FaceDetectQA end "+ String.*valueOf*(result));

////////////////////////////////////////////////////////////////////

// 결과점수가 0보다 크면 FaceDetectBuf 의 값을 사용한다.

//////////////////////////////////////////////////////////////////////

if( (result >= 1 ) && (result <= 100) )

{

Toast.*makeText*(act.getBaseContext(), "인식 가능 검출 결과 " + String.*valueOf*(result), Toast.*LENGTH\_LONG*).show();

// 필요시 String 으로 변환

String bufStr = "";

String bufStr2 = "";

for( int i =0 ; i < 3000 ; i++)

{

bufStr = bufStr + Character.*toString*( (char) FaceDetectBuf[i] ) ;

}

for( int i =3000 ; i < 4668 ; i++)

{

bufStr2 = bufStr2 + Character.*toString*( (char) FaceDetectBuf[i] ) ;

}

Log.*i*("FA\_Demo", "특징점 점수="+String.*valueOf*(result));

Log.*i*("FA\_Demo", "특징점="+bufStr); // log 에 한번에 4668 바이트 출력이 안되어서 3000 + 1668 로 나누어서 출력 함

Log.*i*("FA\_Demo", "특징점="+bufStr2);

Toast.*makeText*(act.getBaseContext(), bufStr , Toast.*LENGTH\_LONG*).show();

}

else {

switch(result){

case -20001:

Toast.*makeText*(act.getBaseContext(), "사진품질:초점흐림", Toast.*LENGTH\_LONG*).show();

break;

case -20002:

Toast.*makeText*(act.getBaseContext(), "사진품질:반사광", Toast.*LENGTH\_LONG*).show();

break;

case -20003:

Toast.*makeText*(act.getBaseContext(), "사진품질:홀로그램", Toast.*LENGTH\_LONG*).show();

break;

case -20004:

Toast.*makeText*(act.getBaseContext(), "사진품질:흑백", Toast.*LENGTH\_LONG*).show();

break;

case -401:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:사진 포멧 에러", Toast.*LENGTH\_LONG*).show();

break;

case -201:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:메모리 확보 에러", Toast.*LENGTH\_LONG*).show();

break;

default:

Toast.*makeText*(act.getBaseContext(), "얼굴검출:얼굴 검출 실패", Toast.*LENGTH\_LONG*).show();

break;

}

}

}

}

);

}

}