

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

----------

**BÁO CÁO MÔN HỌC : PROJECT 2**

**ĐỀ TÀI: Face Detection and Recognition Application in Android**

**Giảng viên hướng dẫn**: **TS.Lã Thế Vinh**

**Sinh viên thực hiện: Nguyễn Ngọc Bình**

**MSSV: 20160370**

**Lớp: CNTT1.01.K61**

**Mục lục**

[Lời mở đầu: 3](#_Toc9870375)

[1. Tạo Application với Android Studio có hỗ trợ C++ 4](#_Toc9870376)

[2. Import module OpenCV cho java 5](#_Toc9870377)

[3. Thêm thư viện Dlib vào project 7](#_Toc9870378)

[4. Thực hiện viết ứng dụng nhận diện khuôn mặt trên Android 8](#_Toc9870379)

[Kết quả: 10](#_Toc9870380)

[Link Project: 11](#_Toc9870381)

[Nhận xét: 11](#_Toc9870382)

# 

# Lời mở đầu:

Đầu tiên, em xin chân thành cảm ơn thầy giáo Ts.Lã Thế Vinh đã giúp đỡ và hướng dẫn em hoàn thành project này. Trong quá trình học tập, nghiên cứu, cũng như là trong quá trình làm bài báo cáo, khó tránh khỏi sai sót, rất mong thầy bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của thầy, cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn trong các môn học sau.

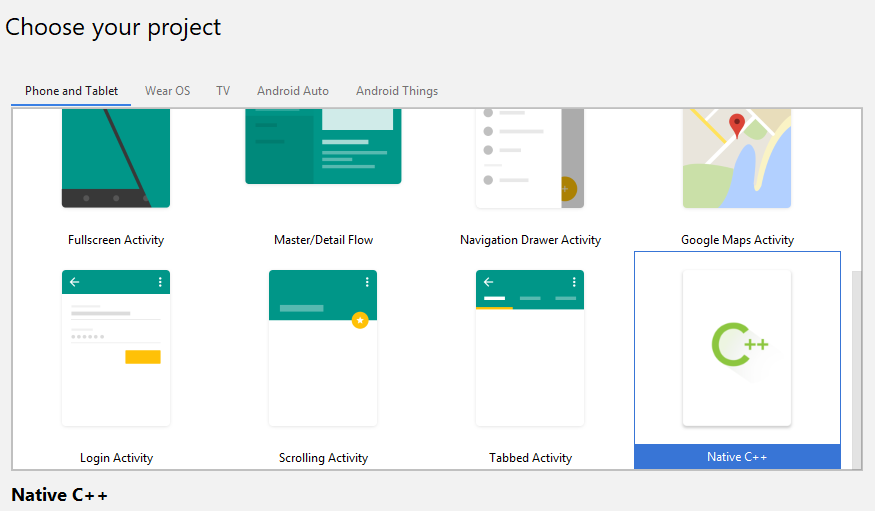
Tóm tắt về Project 1: Project 1 đã thực hiện phân loại khuôn mặt thành công giữa một người là bản thân mình và người khác không phải là mình với độ chính xác khá tốt trên C++ với sự hỗ trợ của OpenCV và Dlib

Bước sang Project 2 này sẽ áp dụng các kiến thức cũng như kết quả ở trên để xây dựng ứng dụng Android dùng trong nhận diện và nhận dạng khuôn mặt tuyến tính, cho phép phân loại được khuôn mặt của mình và không phải của mình

Trong bài viết này tôi giả sử các bạn đã đọc qua project 1 để hiểu cơ cấu của nhận diện khuôn mặt vì tôi sẽ không đi sâu vào phần code trên C++ nữa và trên máy các bạn có cài đặt sẵn AndroidStudio dùng làm IDE cho project.

# 1. Tạo Application với Android Studio có hỗ trợ C++

Chọn menu File 🡪 New 🡪 New Project 🡪 Native C++



Để mặc định là Toolchain Default và click Finish

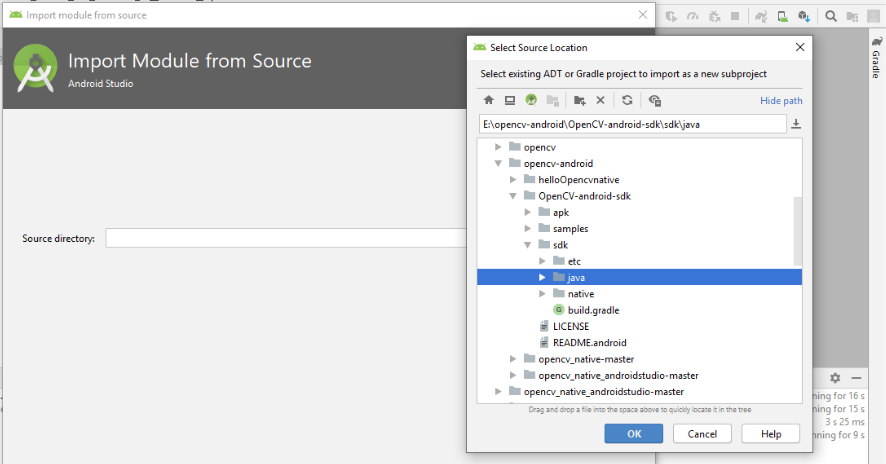
Bây giờ chương trình đã có thể gọi các native code từ C++ để hỗ trợ cho java

# 2. Import module OpenCV cho java

Bước 1: Truy cập vào trang web của OpenCV sdk <https://opencv.org/releases/> để tải bản mới nhất của OpenCV cho Android Studio, sau đó giải nén đến thư mục lưu trữ, chẳng hạn như E:/opencv-android/OpenCV-android-sdk.

Bước 2: Vào menu File 🡪 New 🡪 Import Module

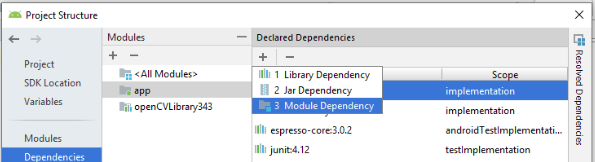
Chọn vào thư mục java theo đường dẫn như trong hình:



Click OK để nguyên 3 tùy chọn đã được tích mặc định và chọn Finish

Đợi Gradle sync xong ta sẽ thấy một số lỗi về các phiên bản SDK được sử dụng trong tệp build.gradle của module OpenCV, chỉ cần sửa lại cho trùng khớp với build.gradle của module app là sẽ hết lỗi. Ngoài ra cần xóa dòng cuối cùng dùng để chỉ định phiên bản minSDK và compileSDK trong file AndriodManifest.xml của module OpenCV.

Bước 3: Vào menu File 🡪 Project Structure 🡪 Dependencies 🡪 module app 🡪 Click vào dấu “+” chọn Module Dependency



Chọn OK, tích chọn vào OpenCVLibrary và chọn tiếp OK

Bước 4: Tạo một folder có tên jniLibs tại thư mục main trong module app/src của Project đang sử dụng. Sau đó vào thư mục libs theo đường dẫn E:\opencv-android\OpenCV-android-sdk\sdk\native\libs. Copy hết các mục trong thư mục libs và dán vào thư mục jniLibs vừa tạo.

Bước 5: Chỉnh sửa CMakeList.txt để có thể compile được thư viện OpenCV từ C++:

Sau dòng cmake\_minimum\_required, thêm vào các dòng sau:

include\_directories(**E:/opencv-android/OpenCV-android-sdk/sdk/native/jni/include**)  
add\_library( **lib\_opencv SHARED IMPORTED** )  
set\_target\_properties(**lib\_opencv PROPERTIES IMPORTED\_LOCATION ${CMAKE\_CURRENT\_SOURCE\_DIR}/../jniLibs/${ANDROID\_ABI}/libopencv\_java3.so**)

Sau dòng native-lib trong mục target-link-library, thêm vào dòng: **lib\_opencv**

Bước 6: Chỉnh sửa build.gradle trong module app

AddabiFilters

externalNativeBuild {  
 cmake {  
 cppFlags **"-frtti -fexceptions"** abiFilters **'x86'**, **'x86\_64'**, **'armeabi-v7a'**, **'arm64-v8a'** arguments **"-DCMAKE\_BUILD\_TYPE=Release"** }  
}

Add OpenCV jniLibs directory:

sourceSets {

main {

jniLibs.srcDirs = ['src/main/jniLibs']

}

}

# 3. Thêm thư viện Dlib vào project

Bước 1: Tải và giải nén thư viện Dlib phiên bản mới nhất từ website <http://dlib.net/> vào module app của Project. Chúng ta chỉ cần thư mục dlib nên có thể xóa các mục còn lại trong thư mục, của mình là Dlib-19.16 để làm giảm kích thước chương trình.

Bước 2: Chỉnh sửa CmakeList.txt để compile thư viện dlib

Sau dòng set\_target\_properties them vào các dòng sau:

set(**DLIB\_DIR ../../../dlib19.16**)  
include\_directories(**${DLIB\_DIR} include**)  
set(**CMAKE\_CXX\_FLAGS** "**${CMAKE\_CXX\_FLAGS} -std=gnu++11**")

Thêm vào các dòng sau ở trong mục add-library

**com\_binhnn\_project2\_facerecognition\_Process.h  
native-lib.cpp  
${DLIB\_DIR}/dlib/all/source.cpp**

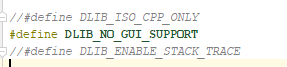
Hai dòng đầu là hai file viết bằng ngôn ngữ C++ dùng để liên kết với java trong Android

Dòng thứ 3 là đường dẫn đến file source.cpp trong dlib

Thêm vào hai dòng sau ở cuối file chỉ định gradle build ở chế độ Release

set(**CMAKE\_C\_FLAGS\_RELEASE** "**${CMAKE\_C\_FLAGS\_RELEASE} -s -O3 -Wall**")  
set(**CMAKE\_CXX\_FLAGS\_RELEASE** "**${CMAKE\_CXX\_FLAGS\_RELEASE} -s -O3 -Wall**")

Khi build nếu có lỗi liên quan đến #include<X11/…> vào file dlib/config.h bật cờ **DLIB\_NO\_GUI\_SUPPORT** bằng cách bỏ comment ở dòng:



# 4. Thực hiện viết ứng dụng nhận diện khuôn mặt trên Android

Đầu tiên trong file AndroidManifest.xml thêm các phụ thuộc cho Camera và bộ nhớ như sau:

<**uses-permission android:name="android.permission.CAMERA"** />  
<**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"**/>

Trong file activity\_main.xml thêm thành phần JavaCameraView của OpenCV:

<**org.opencv.android.JavaCameraView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/javaCameraView"** />

Cấu trúc chương trình như sau:

Java Process.java

MainActivity.java

C++ com\_binhnn\_project2\_facerecognition\_Process.h

Native-lib.cpp

Trong đó:

MainActivity.java dùng để khởi tạo Camera và truyền hình ảnh từ thu được vào class Process.java để xử lý. Sau khi nhận được ảnh, tiến hành resize kích thước xuống còn 320 x ? nhằm mục đích tăng tốc trong quá trình xử lý bức ảnh. Truyền địa chỉ của bức ảnh đã resized xuống lớp C++ để tiến hành nhận diện

**static** {  
 System.*loadLibrary*(**"native-lib"**);  
}  
**public static** Mat processingImage(Mat inputMat){  
 *processingMat* = inputMat;  
 Mat resizedMat = **new** Mat();  
 *scale* = 320.f/Math.*max*(inputMat.width(), inputMat.height());

Size sz = **new** Size((**int**)(*scale*\*inputMat.width()),(**int**)(*scale*\*inputMat.height()));  
Imgproc.*resize*(inputMat, resizedMat, sz);  
**int**[] result = *detectFace*(resizedMat.getNativeObjAddr());  
**if** (result== **null**)  
 **return null**;  
**else** *processingMat* = *displayRecognition*(result);  
 **return** *processingMat*;

}

Hàm detectFace được gọi thông qua phương thức

**public native static int**[] detectFace(**long** addrInput);

Phương thức này sẽ gọi trực tiếp hàm xử lý ảnh từ C++ như sau:

**extern "C"  
JNIEXPORT** jintArray **JNICALL** Java\_com\_binhnn\_project2\_1facerecognition\_Process\_detectFace(JNIEnv \*env, jclass type, jlong addrInput) {  
 cv::Mat& inputMat = \*(cv::Mat\*)addrInput;  
 **int** \*result = faceRecognition(inputMat);  
 **if**(result == **nullptr**)  
 **return nullptr**;  
 **else** {  
 jintArray jresult = env->NewIntArray(len );  
 env->SetIntArrayRegion(jresult, 0, len, &result[0]);  
 **return** jresult;  
 }  
}

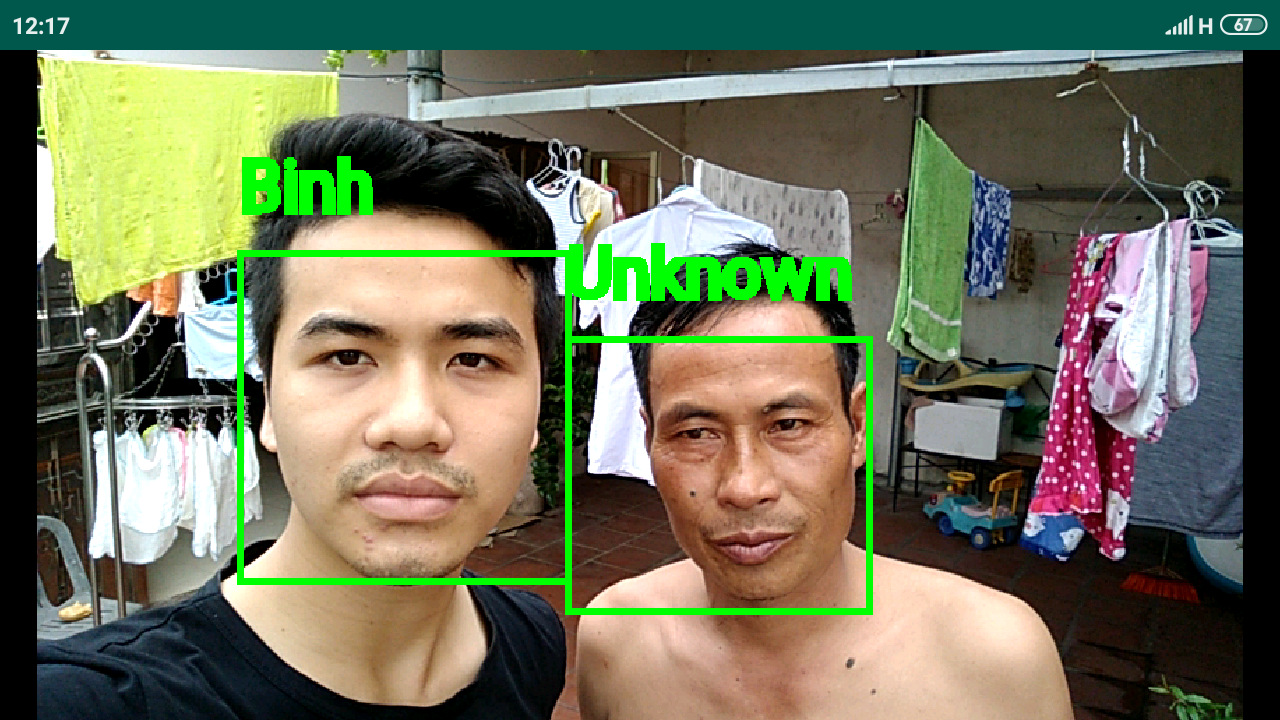
Hàm này được gọi từ C++ có đầu vào là địa chỉ của bức ảnh được gửi xuống, tiến hành load các models và xử lý nhận diện các khuôn mặt có trong bức ảnh, lưu tọa độ của hai điểm trái trên, phải dưới của bounding box bao quanh khuôn mặt cùng với nhãn của nó lưu vào trong một mảng jresult. Nếu đúng là mình thì sẽ có nhãn 1 còn ngược lại thì sẽ có nhãn là 0.

Sau khi nhận được mảng trả về từ C++,

Hàm **public static** Mat displayRecognition(**int**[] result) sẽ tiến hành tách tọa độ của các khuôn mặt cùng với nhãn của chúng, nhân với tỉ lệ để trở thành tọa độ của ảnh ban đầu và hiển thị kết quả nhận diện lên trên màn hình.

Kết quả trên màn hình sẽ bao gồm Bounding box của các khuôn mặt cùng với nhãn. Nếu là tôi thì sẽ có nhãn là “Binh” và ngược lại thì sẽ là “Unknown”

# Kết quả:



# Link Project: <https://github.com/BinhNguyen231/Face-Recognition-Android>

# Nhận xét:

* Tuy đã đạt được kết quả nhận diện và nhận dạng với độ chính xác khá cao nhưng vẫn còn hạn chế về mặt tốc độ xử lý, chỉ vào khoảng 2 đến 3 giây 1 frame.
* Tiếp tục cải tiến để tăng tốc độ xử lý của ứng dụng để đạt kết quả tốt hơn