BÁO CÁO CÔNG VIỆC

[**1. Tổng quan 1**](#_ht1f6vn6g1ck)

[**2. Huấn luyện và nhận diện 2**](#_fw5k6fn1j31p)

[2.1. Hàm LoadEncodeFile 2](#_dthuazgot11b)

[2.2. Hàm Encode 3](#_4n6hbgzezqav)

[2.3. Hàm Recognize 3](#_xh97mra44gbl)

[2.4. Hàm Main 5](#_y09p7a2pnhmv)

[**3. TrainAndRecognition 5**](#_mkq5dbqvigpr)

[3.2. Biến thành viên: 6](#_qqovvlpr0kpf)

[3.2. Các phương thức 6](#_1gjppb367xj)

[3.2.1. Hàm Khởi tạo (TrainAndRecognition): 6](#_43e0rvm9wivq)

[3.2.2. Hàm RunPythonScript: 6](#_6pt6848g9tzy)

[3.2.3. Hàm RunRecognition: 7](#_pcz91mbxwltn)

[**4. FileServer 8**](#_xb1uywz64e6o)

[**5. Phía Client (Android) 10**](#_p77lnid2ynq7)

[5.1. Lớp ImageRotate 10](#_oqpgexeziyd9)

[5.1.1. Các Thành phần Chính 11](#_8fuk0res4jv1)

[5.1.2. Mã nguồn 11](#_hljfypvejonn)

[5.2. Cập nhật lại mã nguồn của CameraxActivity.xml 13](#_t82d9alrd8nf)

[5.3. Cập nhật mã nguồn CameraActivity.java 15](#_9sco2ldzrcj4)

[5.4. Cập nhật mã nguồn FileTransferTask.java 22](#_6j7qtiwfxcmd)

[**6. Kết quả đạt được 26**](#_m0n1qnd46fui)

# 1. Tổng quan

* Công việc số: **45**
* Tên công việc: **Tích hợp module nhận diện vào server C# để nhận diện hình ảnh nhận được từ điện thoại và gửi thông điệp (rectangle, tên người) sang điện thoại, điện thoại vẽ boundingbox và tên người lên ảnh**
* Người thực hiện: **Hạ Quang Dũng**
* Ngày bắt đầu: 06/08/2024
* Ngày kết thúc: 12/08/2024

# 2. Huấn luyện và nhận diện

| import sys import face\_recognition import os from PIL import Image import time |
| --- |

## 2.1. Hàm LoadEncodeFile

**Mô tả**: Đọc một tệp mã hóa và trả về danh sách tên và mã hóa tương ứng.

| def LoadEncodeFile(file\_path):  """  Đọc tệp mã hóa khuôn mặt và trả về danh sách tên và mã hóa tương ứng.    :param file\_path: Đường dẫn đến tệp chứa mã hóa khuôn mặt.  :return: Tuple (names, encodings) nơi names là danh sách tên và encodings là danh sách các mã hóa khuôn mặt.  """  names = []  encodings = []   with open(file\_path, 'r') as file:  lines = file.readlines()   current\_name = None  current\_encoding = []   for line in lines:  parts = line.strip().split(': ')   if len(parts) == 2:  if current\_name is not None:  names.append(current\_name)  encodings.append(current\_encoding)   current\_name = parts[0]  current\_encoding = [float(value) for value in parts[1][1:-1].replace(']', '').split()]  elif current\_name is not None:  current\_encoding.extend([float(value) for value in line.strip().replace(']', '').split()])   if current\_name is not None:  names.append(current\_name)  encodings.append(current\_encoding)   return names, encodings |
| --- |

## 2.2. Hàm Encode

**Mô tả:** Mã hóa khuôn mặt từ ảnh và lưu vào tệp mã hóa.

| def Encode(image\_path, encoding\_file, username):  """  Mã hóa khuôn mặt từ ảnh và lưu vào tệp mã hóa.   :param image\_path: Đường dẫn đến ảnh cần mã hóa.  :param encoding\_file: Đường dẫn đến tệp nơi lưu mã hóa khuôn mặt.  :param username: Tên người dùng để gán cho mã hóa khuôn mặt.  """  if not os.path.isfile(encoding\_file):  with open(encoding\_file, 'w', encoding='utf-8') as file:  file.write('')   with open(encoding\_file, 'a', encoding='utf-8') as encoding\_file:   image = face\_recognition.load\_image\_file(image\_path)  encoding = face\_recognition.face\_encodings(image)   if len(encoding) > 0:  encoding\_file.write(f"{username}: {encoding[0]}\n")  # os.remove(image\_path) # Nếu cần, có thể xóa ảnh sau khi xử lý.  print("Đã xử lý xong") |
| --- |

## 2.3. Hàm Recognize

**Mô tả:** Nhận diện khuôn mặt từ ảnh và ghi kết quả vào tệp.

| def Recognize(image\_path, names, encodings, resultPath):  """  Nhận diện khuôn mặt từ ảnh và ghi kết quả vào tệp.   :param image\_path: Đường dẫn đến ảnh cần nhận diện.  :param names: Danh sách tên người đã mã hóa.  :param encodings: Danh sách mã hóa khuôn mặt tương ứng với tên.  :param resultPath: Đường dẫn đến tệp nơi lưu kết quả nhận diện.  """  image = face\_recognition.load\_image\_file(image\_path)  face\_locations = face\_recognition.face\_locations(image)  face\_encodings = face\_recognition.face\_encodings(image, face\_locations)   results = []  for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):  matches = face\_recognition.compare\_faces(encodings, face\_encoding, tolerance=0.3)  name = "Unknown"   if True in matches:  first\_match\_index = matches.index(True)  name = names[first\_match\_index]   results.append({  "top": top,  "right": right,  "bottom": bottom,  "left": left,  "name": name  })   try:  with open(resultPath, "w", encoding="UTF-8") as file:  if len(results) != 0:  try:  if len(results) > 1:  recognized\_names = [result['name'] for result in results]  file.write("Phát hiện 2 khuôn mặt, vui lòng thử lại!")  else:  file.write(f"{results[0]['name']} ({results[0]['top']}, {results[0]['right']}, {results[0]['bottom']}, {results[0]['left']})")  except Exception as e:  file.write(f"Lỗi khi thực thi script Python: {e}")  else:  file.write("Không có khuôn mặt được tìm thấy\n")  except Exception as e:  with open(resultPath, "w", encoding="UTF-8") as file:  file.write(f"Lỗi khi thực thi script Python: {e}\n") |
| --- |

## 2.4. Hàm Main

**Mô tả:** Phân tích các tham số đầu vào và gọi các hàm tương ứng.

| if \_\_name\_\_ == "\_\_main\_\_":  """  Điểm vào chính của chương trình. Xử lý các lệnh và tham số đầu vào từ dòng lệnh để gọi các hàm tương ứng.  """  if len(sys.argv) < 2:  print("Usage: python RecognizeFace.py command [options]")  sys.exit(1)   command = sys.argv[1]   if command == "Recognize":  image\_path = sys.argv[2]  encodings\_path = sys.argv[3]  result\_path = sys.argv[4]  names, encodings = LoadEncodeFile(encodings\_path)  Recognize(image\_path, names, encodings, result\_path)  elif command == "Recognition2":  image\_path = sys.argv[2]  encodings\_path = sys.argv[3]  names, encodings = LoadEncodeFile(encodings\_path)  Recognize2(image\_path, names, encodings)  elif command == "Encode":  image\_path = sys.argv[2]  encoding\_file = sys.argv[3]  username = sys.argv[4]  Encode(image\_path, encoding\_file, username)  else:  print("Unknown command")  sys.exit(1) |
| --- |

# 3. TrainAndRecognition

**Tên lớp:** TrainAndRecognition

**Chức năng chính:** Lớp TrainAndRecognition thực hiện việc gọi một script Python từ C# để nhận diện khuôn mặt, đồng thời xử lý kết quả trả về từ script và quản lý các đường dẫn liên quan đến các tệp tin và script.

**Bao gồm:**

* Khởi tạo các đường dẫn cần thiết cho script Python và các tệp tin dữ liệu.
* Phương pháp để chạy script Python và thu thập kết quả.
* Phương pháp để nhận diện khuôn mặt từ ảnh và xử lý kết quả.

## 3.2. Biến thành viên:

* \_rootDirectory: Thư mục gốc của ứng dụng.
* \_pythonScriptPath: Đường dẫn đến script Python chính để thực hiện nhận diện.
* \_encodingPath: Đường dẫn đến tệp tin chứa các mã hóa khuôn mặt.
* \_resultPath: Đường dẫn đến tệp tin nơi lưu trữ kết quả nhận diện.

## 3.2. Các phương thức

### 3.2.1. Hàm Khởi tạo (TrainAndRecognition):

**Mô tả:** Xác định các đường dẫn cần thiết cho các tệp tin và script bằng cách xây dựng đường dẫn tương đối từ thư mục gốc của ứng dụng.

**Chi tiết thực hiện:**

* Sử dụng Directory.GetParent để xác định đường dẫn gốc của ứng dụng.
* Tạo đường dẫn đến các tệp tin cần thiết cho việc nhận diện và kết quả.

| public TrainAndRecognition() {  \_rootDirectory = Directory.GetParent(AppContext.BaseDirectory)?.Parent?.Parent?.Parent?.FullName ?? string.Empty;  \_pythonScriptPath = Path.Combine(\_rootDirectory, "Services", "Main.py").Replace('\\', '/');  \_encodingPath = Path.Combine(\_rootDirectory, "Models", "encodings.txt").Replace('\\', '/');  \_resultPath = Path.Combine(\_rootDirectory, "Services", "results.txt").Replace('\\', '/'); } |
| --- |

### 3.2.2. Hàm RunPythonScript:

**Mô tả:** Thực hiện chạy script Python với các tham số đã cung cấp và trả về kết quả đầu ra của script.

**Chi tiết thực hiện:**

* Cấu hình ProcessStartInfo để chạy lệnh Python và thu thập đầu ra.
* Khởi động quá trình, đọc đầu ra từ StandardOutput, và trả về kết quả.

| private string RunPythonScript(string cmd, string args) {  var startInfo = new ProcessStartInfo  {  FileName = cmd,  Arguments = args,  UseShellExecute = false,  RedirectStandardOutput = true,  CreateNoWindow = true  };   using (var process = Process.Start(startInfo))  {  if (process == null) return string.Empty;  using (var reader = process.StandardOutput)  {  string result = reader.ReadToEnd();  Console.WriteLine(result);  return result;  }  } } |
| --- |

### 3.2.3. Hàm RunRecognition:

**Mô tả:** Chạy script Python để nhận diện khuôn mặt từ ảnh, đo thời gian thực thi, và đọc kết quả từ tệp tin.

**Chi tiết thực hiện:**

* Gọi RunPythonScript với lệnh Python cần thiết để thực hiện nhận diện.
* Đo thời gian thực thi hàm và in ra thời gian đã trôi qua.
* Đọc nội dung từ tệp tin kết quả và trả về nội dung đó.

| public string RunRecognition(string imagePath) {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();  RunPythonScript("python", $"\"{\_pythonScriptPath}\" Recognize \"{imagePath}\" \"{\_encodingPath}\" \"{\_resultPath}\"");  stopwatch.Stop();  TimeSpan elapsedTime = stopwatch.Elapsed;  Console.WriteLine($"Thời gian thực thi hàm Run ({elapsedTime.TotalMilliseconds} ms)");   // Đọc nội dung từ file kết quả  string output = string.Empty;  if (File.Exists(\_resultPath))  {  try  {  output = File.ReadAllText(\_resultPath);  }  catch (Exception ex)  {  Console.WriteLine($"Lỗi khi đọc file kết quả: {ex.Message}");  }  }  else  {  Console.WriteLine("File kết quả không tồn tại.");  }   return output; } |
| --- |

# 4. FileServer

**Chức năng chính:** Lớp này thực hiện việc nhận file từ client qua TCP, lưu file vào thư mục chỉ định, thực hiện nhận diện qua phương pháp TrainAndRecognition, và gửi kết quả nhận diện trở lại client.

Tuy nhiên, phần này đã thực hiện ở công việc trước nên ở công việc hiện tại ta chỉ cần chỉnh sửa lại hàm Start() như bên dưới

**Hàm Start:**

* **Mô tả:** Bắt đầu lắng nghe kết nối từ client, nhận file, lưu file, thực hiện nhận diện và gửi kết quả về client.
* **Chi tiết thực hiện:**
  + **Khởi tạo và bắt đầu TcpListener:** Tạo đối tượng TcpListener để lắng nghe kết nối từ client.
  + **Chấp nhận kết nối:** Chấp nhận kết nối từ client và đọc file từ luồng dữ liệu.
  + **Lưu file:** Ghi dữ liệu vào file trong thư mục đã chỉ định.
  + **Nhận diện:** Gọi phương pháp nhận diện từ đối tượng \_trainAndRecognition.
  + **Gửi kết quả:** Gửi kết quả nhận diện trở lại client.

| public void Start() {  try  {  TcpListener listener = new TcpListener(IPAddress.Any, \_port);  listener.Start();   IPEndPoint localEndPoint = listener.LocalEndpoint as IPEndPoint;  if (localEndPoint != null)  {  IPAddress ipv4Address = GetLocalIPv4Address();  Console.WriteLine($"Server đang lắng nghe trên IP: {ipv4Address} và cổng: {localEndPoint.Port}");  }   while (true)  {  using (TcpClient client = listener.AcceptTcpClient())  {  IPEndPoint remoteEndpoint = client.Client.RemoteEndPoint as IPEndPoint;  if (remoteEndpoint != null)  {  Console.WriteLine($"\nĐã kết nối với client: {remoteEndpoint.Address}:{remoteEndpoint.Port}");  }   using (NetworkStream stream = client.GetStream())  using (StreamReader reader = new StreamReader(stream))  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();  string fileName = reader.ReadLine();  if (string.IsNullOrEmpty(fileName))  {  Console.WriteLine("Tên file không hợp lệ.");  continue;  }  string filePath = Path.Combine(\_directoryPath, fileName);   using (FileStream fileStream = File.Create(filePath))  {  byte[] buffer = new byte[4096];  int bytesRead;  while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)  {  fileStream.Write(buffer, 0, bytesRead);  }  Console.WriteLine($"File {fileName} đã được nhận và lưu vào thư mục.");  }  stopwatch.Stop();   // In ra thời gian đã trôi qua  TimeSpan elapsedTime = stopwatch.Elapsed;  Console.WriteLine($"Thời gian lưu file: {elapsedTime.TotalMilliseconds} ms");   stopwatch.Restart();  stopwatch.Start();  string output = \_trainAndRecognition.RunRecognition(filePath);  stopwatch.Stop();   elapsedTime = stopwatch.Elapsed;  Console.WriteLine($"{output} ({elapsedTime.TotalMilliseconds} ms)");   byte[] response = Encoding.UTF8.GetBytes($"{output}");  stream.Write(response, 0, response.Length);  stream.Flush();  }  }  }  }  catch (Exception ex)  {  Console.WriteLine("Lỗi khi khởi động server: " + ex.ToString());  } } |
| --- |

# 5. Phía Client (Android)

## 5.1. Lớp ImageRotate

**Tên lớp:** ImageRotator

**Chức năng chính:** Lớp ImageRotator cung cấp các phương thức để xoay và lật ảnh. Phương thức chính thực hiện các thao tác này là rotateAndFlipImage.

Lớp ImageRotator thực hiện các nhiệm vụ sau:

* Đọc ảnh từ một URI.
* Xoay ảnh theo một số độ góc nhất định.
* Lật ảnh theo trục Y nếu cần.
* Lưu ảnh đã được xử lý trở lại vào bộ nhớ.

### 5.1.1. Các Thành phần Chính

**Hằng số:**

* TAG: Một chuỗi để ghi nhật ký thông tin lỗi liên quan đến việc xử lý ảnh.

**Phương thức rotateAndFlipImage:**

* **Mô tả:** Xoay và lật ảnh dựa trên các tham số đầu vào và lưu ảnh đã được xử lý trở lại vào bộ nhớ.
* **Tham số:**
  + contentResolver: Đối tượng ContentResolver để đọc và ghi dữ liệu ảnh.
  + uri: URI của ảnh cần xử lý.
  + rotationDegrees: Số độ để xoay ảnh.
  + flipVertically: Boolean chỉ định xem có cần lật ảnh theo trục Y hay không.
* **Chi tiết thực hiện:**
  + **Đọc ảnh từ URI:** Sử dụng BitmapFactory để giải mã ảnh từ URI.
  + **Tạo ma trận:** Sử dụng Matrix để định nghĩa các thao tác xoay và lật ảnh.
  + **Xoay ảnh:** Áp dụng phép xoay theo số độ được cung cấp.
  + **Lật ảnh:** Nếu cần, áp dụng phép lật theo trục Y.
  + **Tạo bitmap mới:** Tạo ảnh mới dựa trên ma trận đã định nghĩa.
  + **Ghi ảnh đã xử lý:** Sử dụng OutputStream để ghi ảnh đã xử lý trở lại vào URI.
  + **Giải phóng bộ nhớ:** Giải phóng bộ nhớ của các bitmap không còn sử dụng.
* **Xử lý lỗi:** Ghi lỗi vào log nếu có vấn đề trong quá trình đọc, xử lý hoặc ghi ảnh.

### 5.1.2. Mã nguồn

| package work.ngangiang.camera.Services;  import android.content.ContentResolver; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.graphics.Matrix; import android.net.Uri; import android.util.Log;  import java.io.IOException; import java.io.OutputStream;  public class ImageRotator {  private static final String TAG = "ImageRotator";   public static void rotateAndFlipImage(ContentResolver contentResolver, Uri uri, int rotationDegrees, boolean flipVertically) {  try {  // Đọc ảnh từ URI  Bitmap bitmap = BitmapFactory.decodeStream(contentResolver.openInputStream(uri));   // Tạo ma trận để xoay ảnh  Matrix matrix = new Matrix();  matrix.postRotate(rotationDegrees);   // Nếu cần lật ảnh theo trục Y  if (flipVertically) {  matrix.postScale(-1, 1, bitmap.getWidth() / 2f, bitmap.getHeight() / 2f);  }   Bitmap transformedBitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);   // Ghi ảnh đã xoay và lật trở lại vào bộ nhớ  try (OutputStream outputStream = contentResolver.openOutputStream(uri)) {  transformedBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);  }   bitmap.recycle();  transformedBitmap.recycle();  } catch (IOException e) {  Log.e(TAG, "Failed to transform image", e);  }  } } |
| --- |

## 5.2. Cập nhật lại mã nguồn của CameraxActivity.xml

Cập nhật thêm nút chuyển đổi camera (Cam trước hoặc sau) như phần in đậm phía dưới

| <?xml version="1.0" encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="@color/black"  tools:context=".CameraxActivity">  <androidx.camera.view.PreviewView  android:id="@+id/viewFinder"  android:layout\_width="0dp"  android:layout\_height="0dp"  android:layout\_marginBottom="10dp"  app:layout\_constraintBottom\_toTopOf="@+id/btnCapture"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="1.0"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/linearLayout"  app:layout\_constraintVertical\_bias="0.0" />  <LinearLayout  android:id="@+id/linearLayout"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:gravity="center\_vertical"  android:weightSum="3"  app:layout\_constraintTop\_toTopOf="parent">  <ImageButton  android:id="@+id/btnFlash"  android:layout\_width="20dp"  android:layout\_height="match\_parent"  android:layout\_alignParentEnd="true"  android:layout\_weight="1"  android:background="?attr/selectableItemBackgroundBorderless"  android:padding="10dp"  android:scaleType="fitCenter"  android:src="@drawable/ic\_flash\_off"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  <TextView  android:id="@+id/btnAspectRatio3\_4"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:paddingVertical="10dp"  android:text="3:4"  android:textAlignment="center"  android:textColor="#fff"  android:textStyle="bold" />  <TextView  android:id="@+id/btnAspectRatio9\_16"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:paddingVertical="10dp"  android:text="9:16"  android:textAlignment="center"  android:textColor="#fff"  android:textStyle="bold" />  </LinearLayout>  <ImageButton  android:id="@+id/btnClose"  android:layout\_width="80dp"  android:layout\_height="80dp"  android:layout\_alignParentEnd="true"  android:layout\_marginStart="10dp"  android:layout\_marginBottom="10dp"  android:background="?attr/selectableItemBackgroundBorderless"  android:padding="20dp"  android:scaleType="fitCenter"  android:src="@drawable/ic\_close"  android:visibility="gone"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintStart\_toStartOf="parent" />  <ImageButton  android:id="@+id/btnSend"  android:layout\_width="80dp"  android:layout\_height="80dp"  android:layout\_alignParentEnd="true"  android:layout\_marginBottom="10dp"  android:background="?attr/selectableItemBackgroundBorderless"  android:padding="6dp"  android:scaleType="fitCenter"  android:src="@drawable/ic\_send"  android:visibility="gone"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" />  <ImageButton  android:id="@+id/btnCapture"  android:layout\_width="70dp"  android:layout\_height="70dp"  android:layout\_alignParentEnd="true"  android:layout\_marginBottom="10dp"  android:background="?attr/selectableItemBackgroundBorderless"  android:scaleType="fitCenter"  android:src="@drawable/btncapture"  android:visibility="visible"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" />  <ImageButton  android:id="@+id/btn\_switch\_camera"  android:layout\_width="50dp"  android:layout\_height="50dp"  android:layout\_marginEnd="30dp"  android:layout\_marginBottom="25dp"  android:background="?attr/selectableItemBackgroundBorderless"  android:scaleType="fitCenter"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:srcCompat="@drawable/ic\_switch\_camera" />  **<ImageView**  **android:id="@+id/imgv"**  **android:layout\_width="wrap\_content"**  **android:layout\_height="0dp"**  **android:layout\_marginBottom="10dp"**  **android:visibility="gone"**  **app:layout\_constraintBottom\_toTopOf="@+id/btnCapture"**  **app:layout\_constraintEnd\_toEndOf="parent"**  **app:layout\_constraintStart\_toStartOf="parent"**  **app:layout\_constraintTop\_toTopOf="@+id/viewFinder"**  **tools:srcCompat="@tools:sample/backgrounds/scenic" />**  </androidx.constraintlayout.widget.ConstraintLayout> |
| --- |

## 5.3. Cập nhật mã nguồn CameraActivity.java

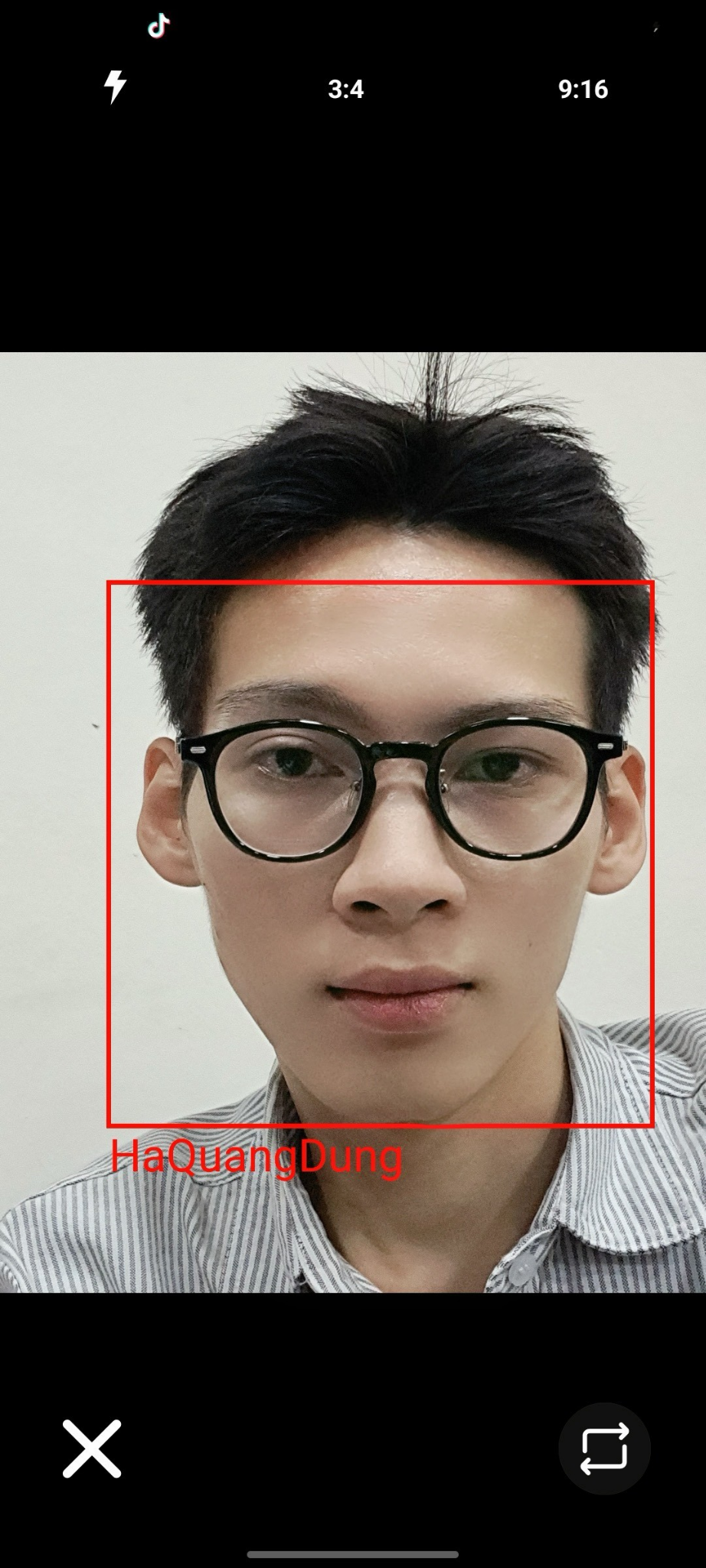
| package work.ngangiang.camera;  import android.Manifest;  import android.annotation.SuppressLint;  import android.content.ContentValues;  import android.content.pm.PackageManager;  **import android.graphics.Bitmap;**  import android.net.Uri;  import android.os.Build;  import android.os.Bundle;  import android.os.Environment;  import android.os.StrictMode;  import android.provider.MediaStore;  import android.util.Log;  import android.view.ScaleGestureDetector;  import android.widget.Button;  import android.widget.ImageButton;  **import android.widget.ImageView;**  import android.widget.LinearLayout;  import android.widget.Toast;  import androidx.activity.EdgeToEdge;  import androidx.annotation.NonNull;  import androidx.appcompat.app.AppCompatActivity;  import androidx.camera.core.AspectRatio;  import androidx.camera.core.Camera;  import androidx.camera.core.CameraControl;  import androidx.camera.core.CameraInfo;  import androidx.camera.core.CameraSelector;  import androidx.camera.core.ImageCapture;  import androidx.camera.core.ImageCaptureException;  import androidx.camera.core.Preview;  import androidx.camera.lifecycle.ProcessCameraProvider;  import androidx.camera.view.PreviewView;  import androidx.core.app.ActivityCompat;  import androidx.core.content.ContextCompat;  import androidx.core.graphics.Insets;  import androidx.core.view.ViewCompat;  import androidx.core.view.WindowInsetsCompat;  import com.google.common.util.concurrent.ListenableFuture;  **import java.io.ByteArrayOutputStream;**  **import java.io.IOException;**  **import java.io.OutputStream;**  import java.util.Objects;  import java.util.concurrent.ExecutionException;  import work.ngangiang.camera.Services.FileTransferTask;  **import work.ngangiang.camera.Services.ImageRotator;**  public class CameraxActivity extends AppCompatActivity {  private static final String TAG = "CameraxActivity";  private static final int REQUEST\_PERMISSIONE = 100;  private PreviewView previewView;  private ImageCapture imageCapture;  private CameraControl cameraControl;  private CameraInfo cameraInfo;  **private ImageButton btnCapture, btnFlash, btnSend, btnClose, btnSwitchCamera;**  **private ImageView imgv;**  **private boolean isFrontCamera = false;**  private int currentAspectRatio = AspectRatio.RATIO\_4\_3;  private ProcessCameraProvider cameraProvider;  private Uri imageUri;  @SuppressLint("MissingInflatedId")  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  EdgeToEdge.enable(this);  setContentView(R.layout.activity\_camerax);  ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  return insets;  });  initViews();  requestAppPermissions();  ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.getInstance(this);  cameraProviderFuture.addListener(() -> {  try {  cameraProvider = cameraProviderFuture.get();  bindPreview();  } catch (ExecutionException | InterruptedException e) {  Log.e(TAG, "Failed to get camera provider", e);  }  }, ContextCompat.getMainExecutor(this));  btnCapture.setOnClickListener(v -> takePhoto());  btnFlash.setOnClickListener(v -> toggleFlash());  **btnClose.setOnClickListener(v -> returnToCamera());**  btnSend.setOnClickListener(v -> {  if (imageUri != null) {  // Gửi file ảnh  **new FileTransferTask(CameraxActivity.this, imgv).execute(imageUri);**  btnSend.setVisibility(Button.GONE);  findViewById(R.id.linearLayout).setVisibility(LinearLayout.VISIBLE);  }  });  **btnSwitchCamera.setOnClickListener(v -> {**  **isFrontCamera = !isFrontCamera;**  **bindPreview();**  **});**  // Bỏ qua lỗi mạng trên main thread (không khuyến khích trong ứng dụng thực tế)  StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  StrictMode.setThreadPolicy(policy);  }  private void initViews() {  previewView = findViewById(R.id.viewFinder);  btnCapture = findViewById(R.id.btnCapture);  btnFlash = findViewById(R.id.btnFlash);  btnSend = findViewById(R.id.btnSend);  btnClose = findViewById(R.id.btnClose);  **btnSwitchCamera = findViewById(R.id.btn\_switch\_camera);**  **imgv = findViewById(R.id.imgv);**  findViewById(R.id.btnAspectRatio3\_4).setOnClickListener(v -> setAspectRatio(AspectRatio.RATIO\_4\_3)); // 3:4  findViewById(R.id.btnAspectRatio9\_16).setOnClickListener(v -> setAspectRatio(AspectRatio.RATIO\_16\_9)); // 9:16  }  private void requestAppPermissions() {  if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.LOLLIPOP &&  (!hasReadPermissions() || !hasWritePermissions() || !hasCameraPermissions())) {  // Yêu cầu các quyền cần thiết, bao gồm quyền camera  ActivityCompat.requestPermissions(this,  new String[]{  Manifest.permission.READ\_EXTERNAL\_STORAGE,  Manifest.permission.WRITE\_EXTERNAL\_STORAGE,  Manifest.permission.CAMERA  }, REQUEST\_PERMISSIONE);  }  }  private boolean hasCameraPermissions() {  return ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED;  }  private boolean hasReadPermissions() {  return ContextCompat.checkSelfPermission(getBaseContext(), Manifest.permission.READ\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED;  }  private boolean hasWritePermissions() {  return ContextCompat.checkSelfPermission(getBaseContext(), Manifest.permission.WRITE\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED;  }  private void bindPreview() {  if (cameraProvider == null) return;  Preview preview = new Preview.Builder()  .setTargetAspectRatio(currentAspectRatio)  .build();  preview.setSurfaceProvider(previewView.getSurfaceProvider());  **CameraSelector cameraSelector = new CameraSelector.Builder()**  **.requireLensFacing(isFrontCamera ? CameraSelector.LENS\_FACING\_FRONT : CameraSelector.LENS\_FACING\_BACK)**  **.build();**  imageCapture = new ImageCapture.Builder()  .setTargetAspectRatio(currentAspectRatio)  .build();  cameraProvider.unbindAll();  Camera camera = cameraProvider.bindToLifecycle(this, cameraSelector, preview, imageCapture);  cameraControl = camera.getCameraControl();  cameraInfo = camera.getCameraInfo();  enablePinchToZoom();  }  private void setAspectRatio(int aspectRatio) {  currentAspectRatio = aspectRatio;  bindPreview();  }  @SuppressLint("ClickableViewAccessibility")  private void enablePinchToZoom() {  ScaleGestureDetector scaleGestureDetector = new ScaleGestureDetector(this, new ScaleGestureDetector.SimpleOnScaleGestureListener() {  @Override  public boolean onScale(@NonNull ScaleGestureDetector detector) {  float currentZoomRatio = Objects.requireNonNull(cameraInfo.getZoomState().getValue()).getZoomRatio();  float delta = detector.getScaleFactor();  cameraControl.setZoomRatio(currentZoomRatio \* delta);  return true;  }  });  previewView.setScaleType(PreviewView.ScaleType.FIT\_CENTER);  previewView.setOnTouchListener((v, event) -> {  scaleGestureDetector.onTouchEvent(event);  return true;  });  }  private void takePhoto() {  ContentValues contentValues = new ContentValues();  contentValues.put(MediaStore.MediaColumns.DISPLAY\_NAME, "photo\_" + System.currentTimeMillis() + ".jpg");  contentValues.put(MediaStore.MediaColumns.MIME\_TYPE, "image/jpeg");  contentValues.put(MediaStore.Images.Media.RELATIVE\_PATH, Environment.DIRECTORY\_PICTURES + "/CameraX");  ImageCapture.OutputFileOptions outputOptions =  new ImageCapture.OutputFileOptions.Builder(  getContentResolver(),  MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI,  contentValues  ).build();  imageCapture.takePicture(outputOptions, ContextCompat.getMainExecutor(this), new ImageCapture.OnImageSavedCallback() {  @Override  public void onImageSaved(@NonNull ImageCapture.OutputFileResults outputFileResults) {  imageUri = outputFileResults.getSavedUri(); // Gán URI vào biến imageUri  if (imageUri != null) {  String msg = "Photo capture succeeded: " + imageUri.toString();  Log.d(TAG, msg);  **// Xoay ảnh sau khi lưu**  **ImageRotator.rotateAndFlipImage(getContentResolver(), imageUri, 270, true);**  **// Giảm dung lượng ảnh**  **reduceImageSize(imageUri);**  } else {  Log.e(TAG, "Photo capture succeeded but URI is null");  }  // Dừng camera sau khi chụp  if (cameraProvider != null) {  cameraProvider.unbindAll();  }  viewSend();  }  @Override  public void onError(@NonNull ImageCaptureException exception) {  String msg = "Photo capture failed: " + exception.getMessage();  Log.e(TAG, msg);  Toast.makeText(getApplicationContext(), msg, Toast.LENGTH\_SHORT).show();  }  });  }  **private void reduceImageSize(Uri imageUri) {**  **try {**  **// Bước 1: Tải ảnh từ URI**  **Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), imageUri);**  **// Bước 2: Giảm kích thước ảnh (giả sử giảm kích thước xuống một nửa)**  **int width = bitmap.getWidth() / 6;**  **int height = bitmap.getHeight() / 6;**  **Bitmap resizedBitmap = Bitmap.createScaledBitmap(bitmap, width, height, true);**  **// Bước 3: Nén ảnh**  **ByteArrayOutputStream outputStream = new ByteArrayOutputStream();**  **// resizedBitmap.compress(Bitmap.CompressFormat.JPEG, 80, outputStream); // Chất lượng 12 (100 là chất lượng gốc)**  **// Bước 4: Ghi đè ảnh đã nén lên ảnh gốc**  **OutputStream imageOutputStream = getContentResolver().openOutputStream(imageUri, "w");**  **if (imageOutputStream != null) {**  **imageOutputStream.write(outputStream.toByteArray());**  **imageOutputStream.close();**  **Log.d(TAG, "Compressed photo overwritten: " + imageUri.toString());**  **}**  **} catch (IOException e) {**  **Log.e(TAG, "Error reducing image size: " + e.getMessage());**  **}**  **}**  private void toggleFlash() {  int flashMode = imageCapture.getFlashMode() == ImageCapture.FLASH\_MODE\_OFF  ? ImageCapture.FLASH\_MODE\_ON  : ImageCapture.FLASH\_MODE\_OFF;  imageCapture.setFlashMode(flashMode);  btnFlash.setImageResource(flashMode == ImageCapture.FLASH\_MODE\_ON  ? R.drawable.ic\_flash\_on  : R.drawable.ic\_flash\_off);  }  private void viewSend() {  btnCapture.setVisibility(Button.GONE);  btnSend.setVisibility(Button.VISIBLE);  btnClose.setVisibility(Button.VISIBLE);  findViewById(R.id.linearLayout).setVisibility(LinearLayout.GONE);  btnFlash.setImageResource(R.drawable.ic\_flash\_off);  }  **public void returnToCamera() {**  **btnCapture.setVisibility(Button.VISIBLE);**  **btnSend.setVisibility(Button.GONE);**  **btnClose.setVisibility(Button.GONE);**  **imgv.setVisibility(ImageView.GONE);**  **bindPreview();**  **findViewById(R.id.linearLayout).setVisibility(LinearLayout.VISIBLE);**  **}**  } |
| --- |

## 5.4. Cập nhật mã nguồn FileTransferTask.java

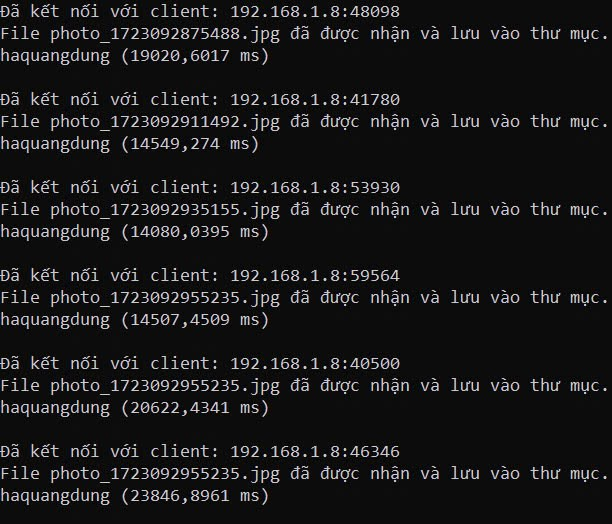
| package work.ngangiang.camera.Services;  import android.content.ContentResolver;  import android.content.Context;  import android.database.Cursor;  **import android.graphics.Bitmap;**  **import android.graphics.BitmapFactory;**  **import android.graphics.Canvas;**  **import android.graphics.Color;**  **import android.graphics.Paint;**  import android.net.Uri;  import android.os.AsyncTask;  **import android.os.Handler;**  **import android.os.Looper;**  import android.util.Log;  import android.widget.ImageView;  import android.widget.Toast;  import java.io.BufferedInputStream;  **import java.io.BufferedOutputStream;**  **import java.io.BufferedReader;**  import java.io.InputStream;  **import java.io.InputStreamReader;**  import java.io.OutputStream;  import java.net.Socket;  **import work.ngangiang.camera.CameraxActivity;**  public class FileTransferTask extends AsyncTask<Uri, Void, String> {  **private Uri FILEURI;**  private static final String SERVER\_IP = "192.168.1.7"; // Địa chỉ IP của server  private static final int SERVER\_PORT = 100; // Cổng của server  private Context context;  **private ImageView imageView;**  **// Constructor để nhận Context và ImageView**  **public FileTransferTask(Context context, ImageView imageView) {**  **this.context = context;**  **this.imageView = imageView;**  **}**  @Override  protected String doInBackground(Uri... uris) {  **Uri fileUri = uris[0];**  **setFILEURI(fileUri);**  String responseMessage = "";  try (  Socket socket = new Socket(SERVER\_IP, SERVER\_PORT);  InputStream inputStream = context.getContentResolver().openInputStream(fileUri);  **OutputStream outputStream = socket.getOutputStream();**  BufferedInputStream bis = new BufferedInputStream(inputStream);  **BufferedOutputStream bos = new BufferedOutputStream(outputStream);**  **BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream(), "UTF-8"));**  ) {  if (inputStream == null) {  Log.e("FileTransferTask", "InputStream is null.");  **return "Luồng dữ liệu không hợp lệ!";**  }  // Lấy tên file từ URI  String fileName = getFileName(fileUri);  // Gửi tên file trước dữ liệu  bos.write((fileName + "\n").getBytes());  bos.flush();  // Gửi dữ liệu file  byte[] buffer = new byte[4096];  int bytesRead;  while ((bytesRead = bis.read(buffer)) != -1) {  bos.write(buffer, 0, bytesRead);  }  bos.flush();  socket.shutdownOutput();  // Nhận thông điệp từ server  responseMessage = reader.readLine();  return responseMessage;  } catch (Exception e) {  e.printStackTrace();  responseMessage = "Gửi file thất bại: " + e.getMessage();  return responseMessage;  }  }  **@Override**  **protected void onPostExecute(String result) {**  **Toast.makeText(context, result, Toast.LENGTH\_SHORT).show();**  **if (result.equals("Không có khuôn mặt được tìm thấy")) {**  **if (context instanceof CameraxActivity) {**  **((CameraxActivity) context).returnToCamera();**  **}**  **return;**  **};**  **// Vẽ bounding box sau khi gửi file**  **if (FILEURI != null) {**  **Bitmap bitmap = getBitmapFromUri(FILEURI);**  **if (bitmap != null) {**  **Bitmap bitmapWithBoundingBox = drawBoundingBox(bitmap, result);**  **imageView.setVisibility(ImageView.VISIBLE);**  **imageView.setImageBitmap(bitmapWithBoundingBox);**  **}**  **}**  **Handler handler = new Handler(Looper.getMainLooper());**  **handler.postDelayed(() -> {**  **if (context instanceof CameraxActivity) {**  **((CameraxActivity) context).returnToCamera();**  **}**  **}, 1000); // 1000 milliseconds = 1 second**  **}**  **private Bitmap drawBoundingBox(Bitmap bitmap, String boundingBoxInfo) {**  **Bitmap mutableBitmap = bitmap.copy(Bitmap.Config.ARGB\_8888, true);**  **Canvas canvas = new Canvas(mutableBitmap);**  **Paint paint = new Paint();**  **// Vẽ hình chữ nhật**  **paint.setColor(Color.RED);**  **paint.setStyle(Paint.Style.STROKE);**  **paint.setStrokeWidth(20);**  **String[] parts = boundingBoxInfo.split("[(), ]+");**  **int y1 = Integer.parseInt(parts[1]);**  **int x1 = Integer.parseInt(parts[2]);**  **int y2 = Integer.parseInt(parts[3]);**  **int x2 = Integer.parseInt(parts[4]);**  **canvas.drawRect(x1, y1, x2, y2, paint);**  **// Vẽ tên**  **paint.setColor(Color.RED);**  **paint.setStyle(Paint.Style.FILL);**  **paint.setTextSize(200);**  **canvas.drawText(parts[0], x2, y2 + 200, paint);**  **return mutableBitmap;**  **}**  private String getFileName(Uri uri) {  ContentResolver contentResolver = context.getContentResolver();  String[] projection = {android.provider.MediaStore.Images.Media.DISPLAY\_NAME};  try (Cursor cursor = contentResolver.query(uri, projection, null, null, null)) {  if (cursor != null && cursor.moveToFirst()) {  int nameIndex = cursor.getColumnIndexOrThrow(android.provider.MediaStore.Images.Media.DISPLAY\_NAME);  return cursor.getString(nameIndex);  }  }  return "unknown.jpg";  }  **public void setFILEURI(Uri FILEURI) {**  **this.FILEURI = FILEURI;**  **}**  } |
| --- |

# 6. Kết quả đạt được

Ta nhận thấy, tốc độ xử lý việc nhận diện nhanh hay chậm phụ thuộc vào kích thước của bức ảnh, và độ chính xác còn phụ thuộc vào cả chất lượng hình ảnh. Tuy nhiên ta cần nhận diện ở mức độ chính xác và nhanh, chính vì vậy giảm kích thước hình ảnh là điều cần thiết.



Kích thước hình ảnh gốc sẽ khoảng 3690x4080 (Kích thước khá lớn) để truyền dữ liệu từ client đến server cũng như việc nhận diện (Mất khoảng 20s để nhận dữ liệu, xử lý yêu cầu nhận diện cũng như phản hồi kết quả cho client).

****

Sau khi giảm kích thước đi khoảng 6 đến 8 lần thì kết quả nhận được có thời gian giảm đáng kể

