studocu

Bao cao PBL5 Thung rac thong minh

PBL4: Đồ án mạng máy tính (Trường Đại học Bách Khoa - Đại học Đà Nẵng)

Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO PBL5

DỰ ÁN KỸ THUẬT MÁY TÍNH

ĐỀ TÀI

Thùng rác thông minh

GIẢNG VIÊN HƯỚNG DẪN: TS. Bùi Thị Thanh Thanh

SINH VIÊN THỰC HIỆN: Đỗ Cao Cường

Hoàng Công Trọng

Vũ Hoàng Tín

Trần Tấn Thịnh

Đà Nẵng, 06/2024

This document is available on studocu

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

TÓM TẮT ĐỒ ÁN

Dự án “Thùng rác thông minh” nhằm giải quyết vấn đề rác thải đô thị và ô nhiễm môi trường ngày càng nghiêm trọng do ý thức phân loại rác kém của người dân.Việc tuyên truyền phân loại rác và thu gom rác theo từng loại đã diễn ra nhiều năm nay nhưng vẫn có số người dân không thực hiện theo.Phương pháp giải quyết bao gồm việc thiết và triển khai hình dạng thùng rác, tích hợp các cảm biến, xây dựng mô hình AI để nhận diện và phân loại rác, phát triển phần mềm quản lý và theo dõi.Kết quả đạt được là hệ thống thùng rác thông minh không chỉ nâng cao hiệu quả phân loại rác tại nguồn, giảm thiểu ô nhiễm, mà còn tăng cường nhận thức và hành vi của cộng đồng về bảo vệ môi trường, góp phần xây dựng đô thị xanh, sạch, đẹp.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Tự đánh giá theo 3 mức (Đã hoàn thành/Chưa hoàn thành/Không triển khai)
Hoàng Công Trọng	- Tìm hiểu thuật toán để huấn luyện mô hình. - Thu thập dữ liệu. - Huấn luyện và tinh chỉnh mô hình AI. - Phát triển Server Django Rest Framework, viết API triển khai với Mobile. - Viết báo cáo	Đã hoàn thành
Đỗ Cao Cường	- Thu thật dữ liệu thật chụp từ ESP32 CAM - Thiết lập giao thức kết nối giữa ESP32 CAM và Server - Thiết lập Websocket để truyền nhận thông tin giữa ESP8266 và Server - Xử lí thông tin sau khi nhận được thông tin rác - Triển khai Backend Server qua NGROK và Websocket qua AWS	Đã hoàn thành
Trần Tấn Thịnh	- Viết phần mềm: Android Kotlin app - Hoàn thiện các chức năng của hệ thống bằng cách hiển thị và điều khiển trên App thông qua API từ Server - Hỗ trợ làm phần cứng	Đã hoàn thành

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

	- Viết báo cáo	
Vũ Hoàng Tín	- Thiết kế mô hình phần cứng. - Lắp ráp mạch - Viết code điều khiển servo - Viết báo cáo	Đã hoàn thành

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

MỤC LỤC

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

MỤC LỤC HÌNH ẢNH

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1. Tổng quan đề tài.

Hiện nay, trên thế giới, đã có nhiều sản phẩm thùng rác thông minh được phát triển nhằm giải quyết vấn đề rác thải và bảo vệ môi trường. Trên thị trường quốc tế, một số mẫu thùng rác thông minh nổi bật như thùng rác BigBelly Solar của Mỹ, sử dụng năng lượng mặt trời để nén rác, giảm tần suất thu gom. Tại Hàn Quốc, sản phẩm CleanCUBE tích hợp công nghệ cảm biến và nén rác tự động, giúp giảm chi phí vận hành và quản lý rác thải hiệu quả. Ở Nhật Bản, thùng rác thông minh của công ty Fujitsu có khả năng nhận diện và phân loại rác thông qua công nghệ AI.

Tại Việt Nam, các sản phẩm thùng rác thông minh cũng đã bắt đầu xuất hiện, tuy nhiên những sản phẩm này vẫn gặp nhiều hạn chế về tính năng phân loại rác tự động và độ bền của thiết bị trong điều kiện thời tiết khắc nghiệt.

1.2. Vấn đề cần giải quyết.

Thùng rác thông minh sẽ phải tích hợp cảm biến để phát hiện mức đầy, mở nắp tự động và gửi thông báo đến hệ thống quản lý, có khả năng nhận biết và phân loại các loại rác khác nhau như rác, nhựa, giấy, kim loại... Điều này giúp tối ưu hóa quá trình tái chế và xử lý rác thải, giảm thiểu lượng rác đi vào bãi chôn lấp và tác động tiêu cực đến môi trường.

1.3. Giải pháp tổng quan.

- **Thiết kế và Xây dựng Thùng rác Thông minh:** Thùng rác có kích thước vừa phải, phù hợp để đặt tại các địa điểm công cộng như trường học, công viên, khu dân cư và khu thương mại. Thùng rác được chia thành nhiều ngăn để chứa các loại rác thải khác nhau bao gồm giấy, nhựa, kim loại…
- **Tích hợp Camera và Hệ thống Xử lý Hình ảnh:** Gắn camera bên trong thùng rác để chụp ảnh rác thải khi người dùng bỏ rác vào. Hệ thống sẽ gửi hình ảnh rác thải lên server để xử lý.
- **Ứng dụng Mô hình AI để Phân loại Rác:** Trên server, chúng tôi triển khai mô hình trí tuệ nhân tạo (AI) để phân tích và nhận diện loại rác từ hình ảnh mà camera ESP32 gửi về
- **Cơ Chế Xoay Ngăn Thùng rác:** Thùng rác được trang bị hệ thống cơ học để xoay các ngăn chứa rác đến đúng vị trí theo kết quả phân loại của mô hình AI

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

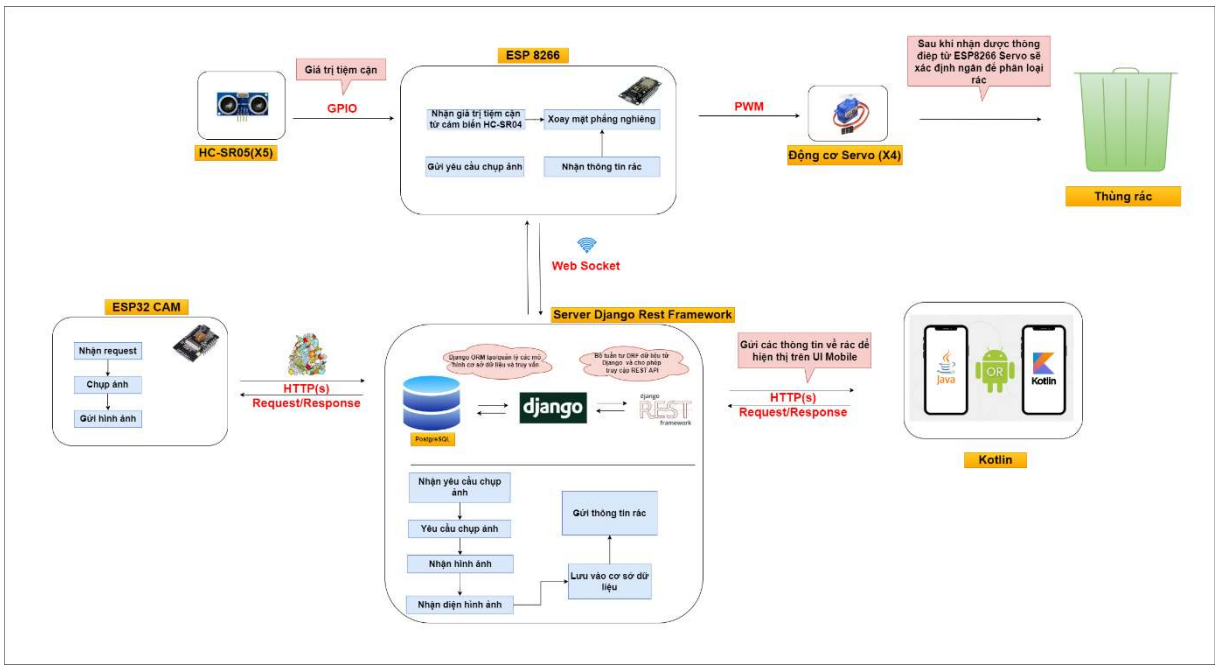
- **Quản lý và Giám sát Thông qua Hệ thống Server:** Hệ thống server không chỉ xử lý và lưu trữ dữ liệu hình ảnh mà còn cung cấp giao diện quản lý, giúp theo dõi tình trạng đầy của các ngăn thùng rác và tối ưu hóa quy trình thu gom rác thải. Thông tin về tình trạng thùng rác sẽ được cập nhật theo thời gian thực, giúp đội ngũ quản lý kịp thời xử lý khi thùng rác đầy

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

CHƯƠNG 2. GIẢI PHÁP THỰC HIỆN

2.1. Sơ đồ khối hệ thống.

2.1.1. Sơ đồ tổng quát của hệ thống:

3. 

Hình 1. Sơ đồ tổng quát của hệ thống.

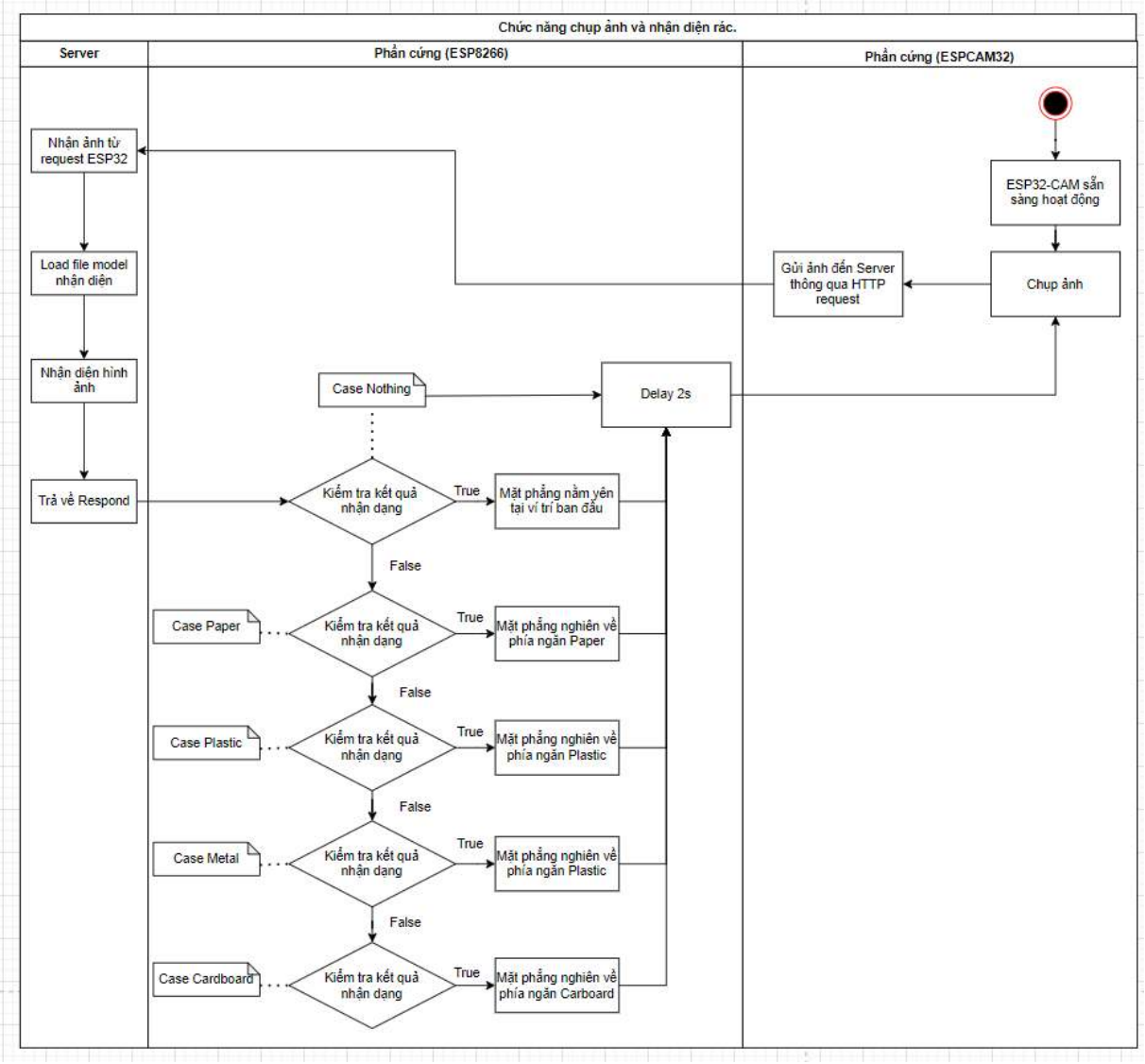
3.1.1. Hệ thống bao gồm:

- ESP32-CAM dùng để chụp ảnh và gửi ảnh về server để nhận diện
- ESP8266 sử dụng để phát hiện người lại gần và xoay mặt phẳng
- Server
- App Mobie để hiển thị, giám sát, giao tiếp và điều khiển hệ thống
- Các thiết bị phần cứng có thể giao tiếp với Server thông qua HTTP (cụ thể là Restful API), và Server giao tiếp với Web thông qua WebSocket. API được lập trình dựa trên Django Rest Framework

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.1.2. Sơ đồ hoạt động các chức năng.

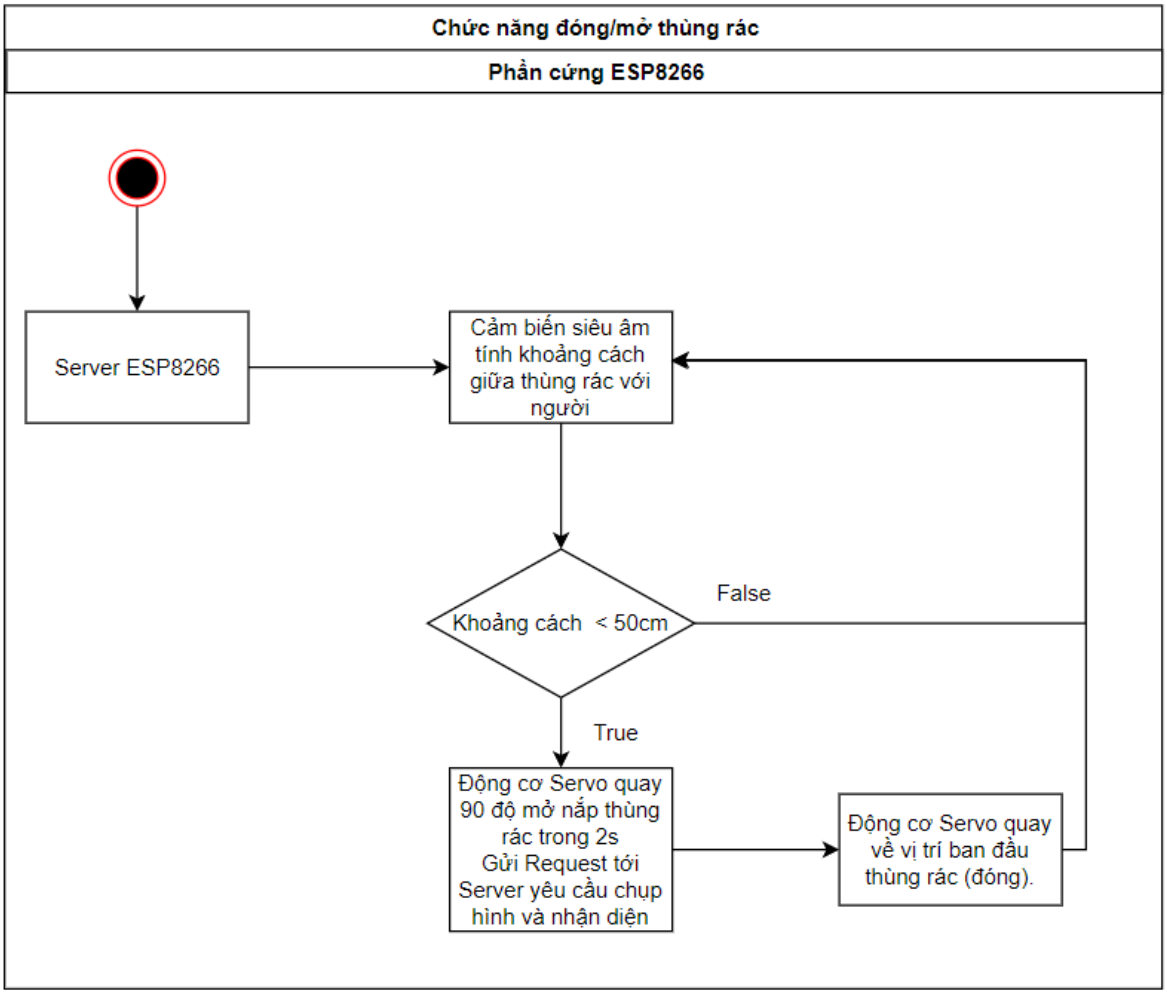
- Chức năng phân loại rác



Hình 2. Sơ đồ hoạt động phân loại rác.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

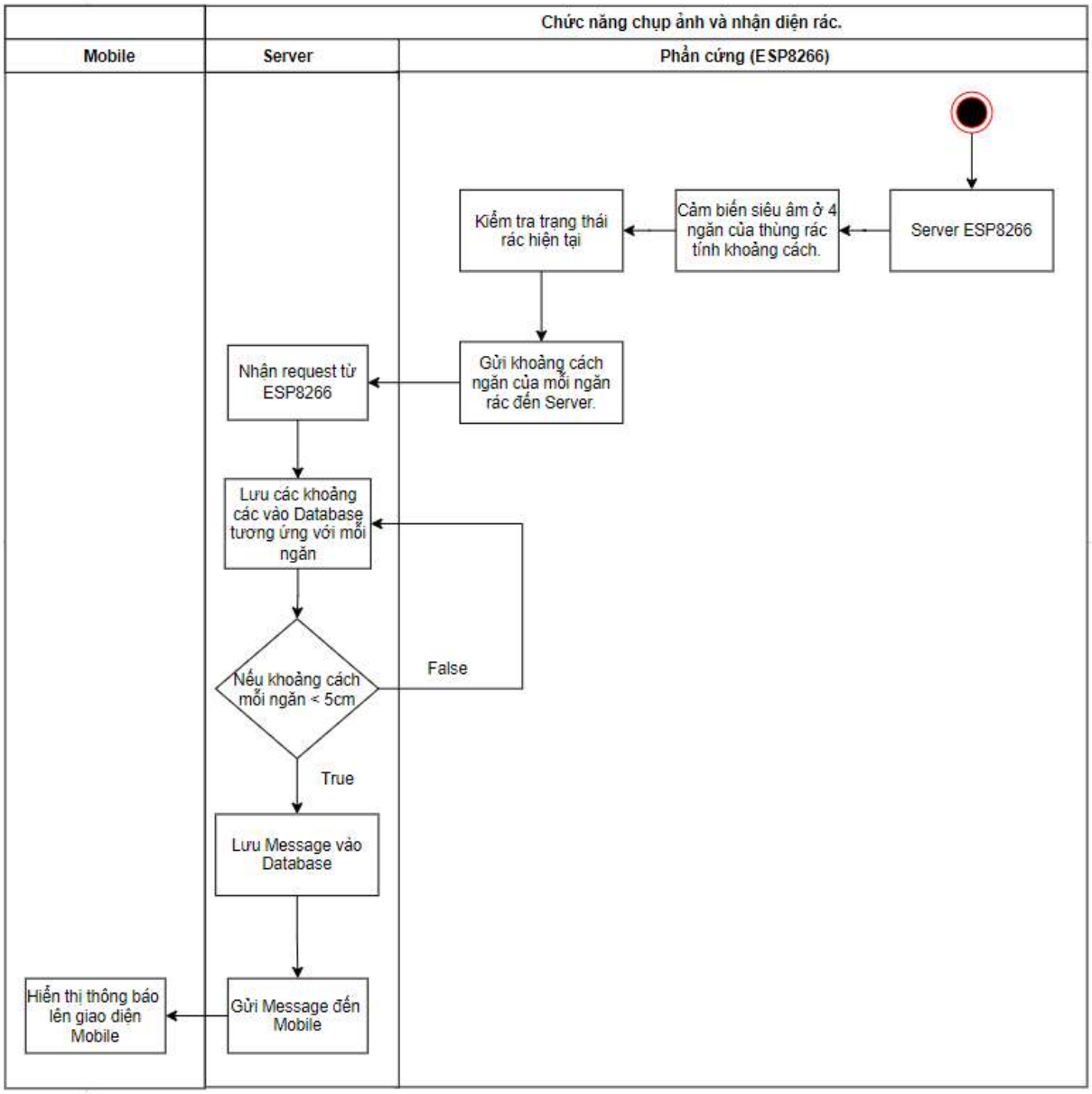
- Chức năng đóng mở thùng rác



Hình 3. Sơ đồ hoạt động đóng mở thùng rác.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Chức năng thông báo thùng rác đầy



Hình 4. Sơ đồ hoạt động chức năng thông báo thùng rác đầy.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.2. Phần cứng và truyền thông.

3.2.1. Danh sách linh kiện.

Tên linh kiện	Chức năng phần cứng	Thông số
ESP32 - CAM x 1	Gửi và nhận dữ liệu chính từ server: - Chụp ảnh rác và gửi lên server	- Điện áp hoạt động: 5V – DC - Cường độ dòng điện: 180mA-310mA - CPU: Xtensa Dual-Core LX6 microprocessor - Bluetooth: v4.2 BR/EDR và BLE - Wi-Fi: 802.11 b/g/n/e/i - RAM: 520 KB SRAM liền chip - Giao tiếp: cổng USB
ESP8266 x 2	Trung tâm xử lí chính của hệ thống, gửi và nhận dữ liệu chính từ server: - Phát hiện người lại gần và gửi yêu cầu lên server - Nhận thông tin rác gửi từ server - Thông qua dữ liệu server gửi, xoay các động cơ servo - Gửi khoảng cách rác để cho server thông tin rác đầy	- Điện áp hoạt động: 5V – DC - GPIO: giao tiếp mức 3.3V - Wi-Fi: 802.11 b/g/n - Chip nạp và console UART: CP2102 - Full IO : 10 GPIO, 1 Analog, 1SPI , 2 UART, 1 I2C/I2S, PWM,… - Bộ nhớ flash: 4MB

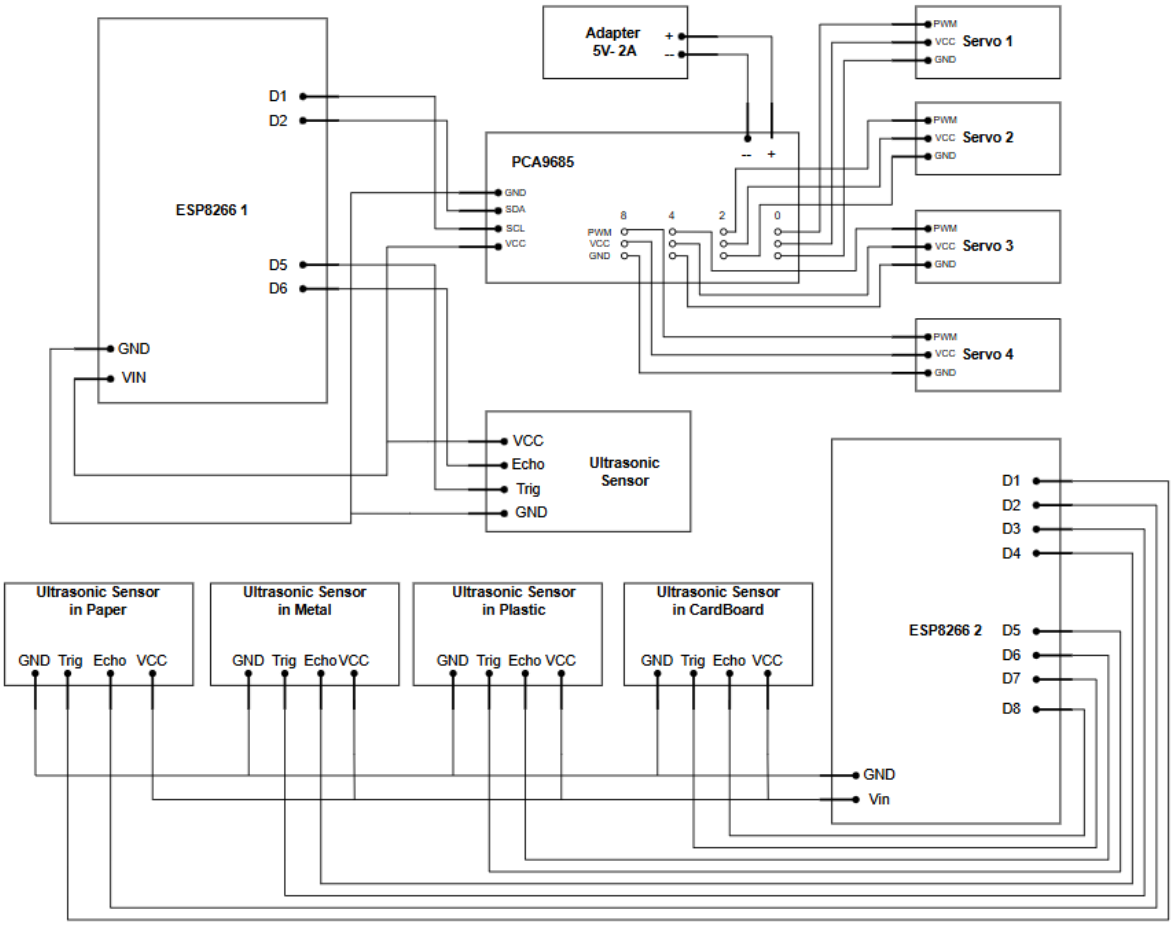
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Cảm biến siêu âm x 5	Đo khoảng cách từ người đến thùng rác:1 Đo khoảng cách từ đáy thúng rác lên mỗi ngăn:4	- Điện áp: 5V DC - Dòng điện hoạt động: < 2mA - Khoảng cách: 2cm – 450cm (4.5m) - Độ chính xác: 3mm
Động cơ servo x 4	Mở đóng thùng rác: 1 Nghiêng mặt phẳng nghiêng:3	- Điện áp hoạt động: 4.8-5V DC - Lực kéo: 1.6 Kg.cm - Độ rộng xung 0.5ms ~ 2.5ms tương ứng 0-180 độ
Mạch điều khiển 16 Channel PWM PCA9685	Kết nối các servo với ESP8266 để điều khiển xoay mặt phẳng	- Điện áp sử dụng: 2.3 ~ 5.5VDC. - Số kênh PWM: 16 kênh,tần số: 40~1000Hz - Độ phân giải PWM: 12bit. - Giao tiếp: I2C (chấp nhận mức Logic TTL 3 ~ 5VDC) - Kích thước: 62.5mm x 25.4mm x 3mm
Dây điện x 50		
Bìa x 4		

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Tên linh kiện	Giá tiền (VNĐ)
ESP32 - CAM x 1	180.000
ESP8266 x 2	150.000
Cảm biến siêu âm x 5	125.000
Động cơ servo x 4	128.000
Bìa x 4	80.000
Dây nối x 50	30.000
Tổng	693.000

3.2.2. Sơ đồ lắp mạch.



Hình 5. Sơ đồ lắp mạch của hệ thống.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

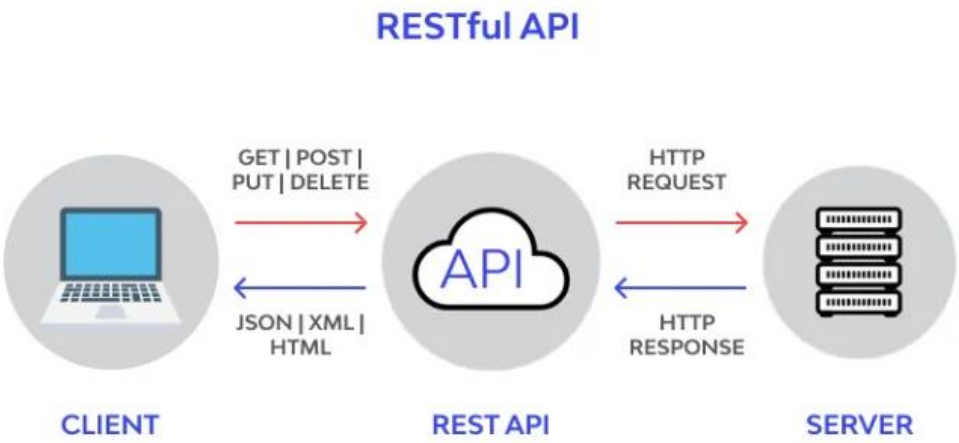
• Nguyên tắc hoạt động mỗi linh kiện:

- **ESP32- CAM**
  - Chức năng cơ bản giống Arduino
  - Có tích hợp một camera cho phép ghi lại hình ảnh hoặc video
  - Có khả năng kết nối Wi-Fi, cho phép gửi và nhận dữ liệu qua mạng Wi-Fi. Nó cũng hỗ trợ Bluetooth, giúp tương tác với các thiết bị khác
  - Chân kết nối là các chân GPIO và hỗ trợ giao diện UART, I2C và SPI
- **ESP8266**
  - Về cơ bản ESP8266 giống ESP32 nhưng không có tích hợp cam
- **Động cơ servo**
  - Code điều khiển động cơ trong thư viện servo.h
  - Động cơ RC servo được điều khiển bằng một tín hiệu điều khiển PWM (Pulse Width Modulation). Tín hiệu này được cung cấp từ một nguồn điều khiển như một bộ điều khiển mô hình RC hoặc mạch điều khiển microcontroller
  - Động cơ RC servo có giới hạn góc quay. Thông thường, góc quay của động cơ servo trong khoảng từ 0 đến 180 độ
- **Cảm biến siêu âm**
  - Phát sóng: Cảm biến siêu âm phát ra một tín hiệu sóng siêu âm từ một bộ phát âm thanh. Sóng siêu âm có tần số cao hơn khả năng nghe thường của con người, thường là khoảng từ 20 kHz đến 200 kHz.
  - Phản xạ: Sóng siêu âm phát ra sẽ lan truyền trong không gian và khi gặp vật thể, nó sẽ bị phản xạ lại.
  - Thu sóng: Cảm biến siêu âm có một bộ thu sóng để nhận sóng siêu âm được phản xạ lại. Bộ thu chuyển đổi sóng siêu âm thành tín hiệu điện.
  - Đo khoảng cách: Dựa trên thời gian mà sóng siêu âm đi từ cảm biến đến vật thể và trở lại, dựa vào thời gian đó tính ra khoảng cách bằng công thức Khoảng cách = (Vận tốc âm thanh trong môi trường x Vận tốc sóng siêu âm x Thời gian bay) / 2 .

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.2.3. Truyền thông

- **Restful API**:



Hình 6. Hoạt động của REST API

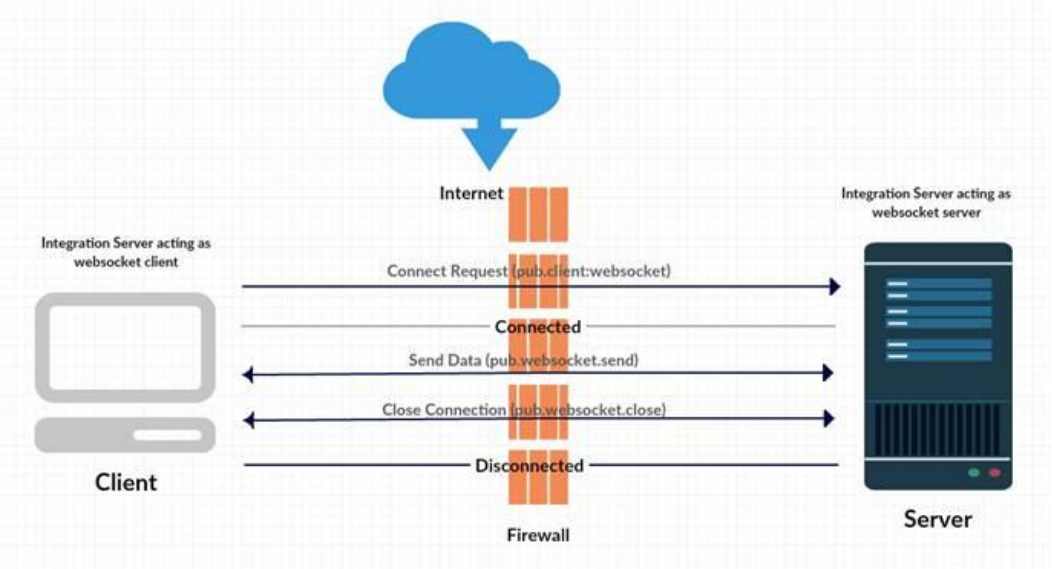
Cách thức hoạt động của Restful API:

RESTful API là phương thức tạo ra API và hoạt động dựa trên phương thức HTTP:

- **GET :** Trả về một Resource hoặc một danh sách Resource.
- **POST** : Tạo mới một Resource.
- **PUT** : Cập nhật thông tin cho Resource
- **DELETE**: Xoá một Resource

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **WebSocket**



Hình 7. Giao thức Websocket

WebSocket là một giao thức truyền tin dựa trên kết nối TCP. WebSocket API cho phép mở phiên giao tiếp tương tác giữa hai chiều giữa trình duyệt của người dùng và máy chủ

Mục đích sử dụng và ưu điểm của WebSocket:

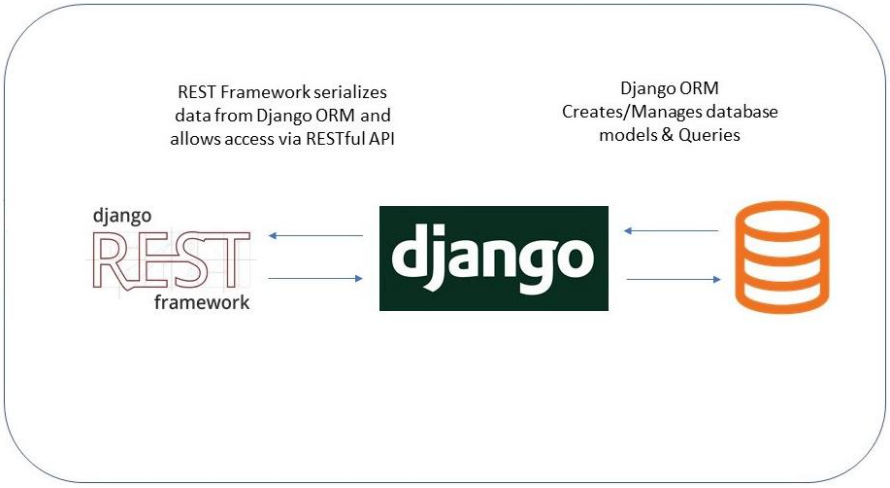
- **WebSocket** cung cấp một kết nối 2 chiều giúp server có thể gửi trực tiếp đến Browser mà không cần đợi yêu cầu từ Browser
- Tốc độ gửi message của WebSocket nhanh chóng giúp thực hiện báo rác đầy một cách nhanh nhất có thể
- Browser chỉ cần handshake kết nối và có thể nhận message báo rác đầy ngay khi ESP8266 gửi yêu cầu

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.3. Phần mềm

3.3.1. Server

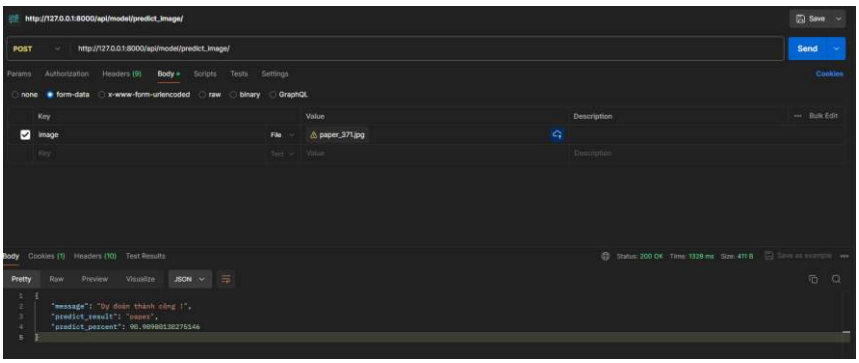
- **Django-Rest-Framework** là một bộ công cụ mạnh mẽ và linh hoạt để xây dựng các API web trong Django. Nó được xây dựng trên Django, một framework nổi tiếng dành cho phát triển ứng dụng web, và cung cấp một tập hợp các tính năng giúp việc xây dựng và duy trì API trở nên dễ dàng và hiệu quả.



Hình 8. Hoạt động của Django Rest Framework

- **API**

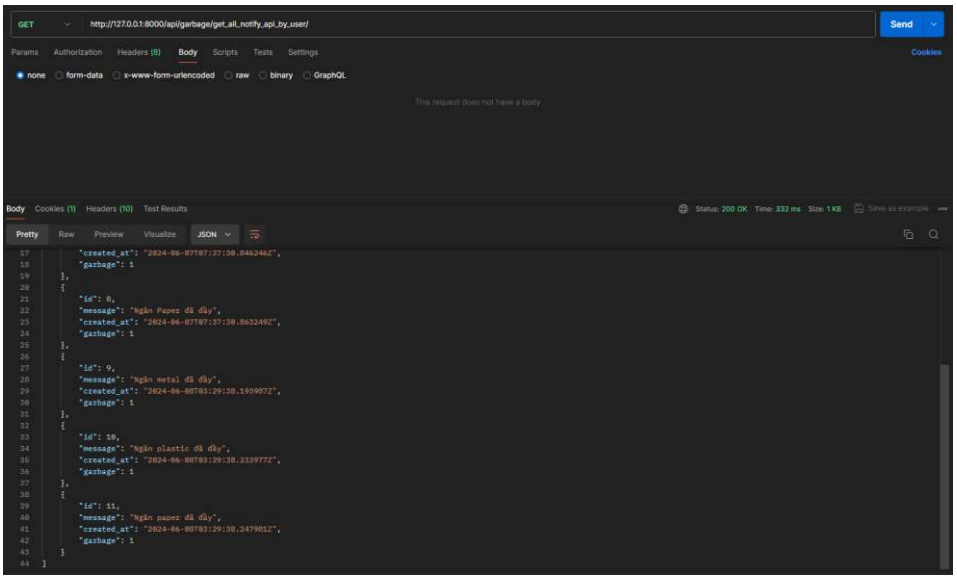
- **Chức năng upload ảnh: ESP-CAM** gửi hình ảnh bằng giao thức HTTP POST với định dạng dữ liệu là Multipart/Form-data bao gồm các tham số **key** "image" và **value** là file ảnh. Response trả về bao gồm các thông tin: message, phần trăm dự đoán, kết quả dự đoán và mã trạng thái status **200 OK.**



Hình 9. Chức năng Upload ảnh ESP32

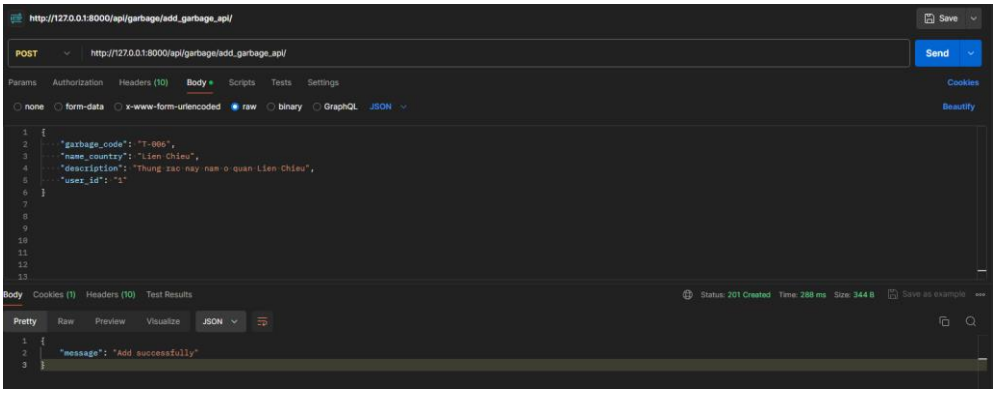
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Chức năng thông báo thùng rác đầy :** Người dùng sẽ đăng nhập và có thể xem thông báo về thùng rác đầy hay chưa, giao thức sử dụng ở đây là HTTP GET. Kết quả trả về là một danh sách các thông báo từ trước đến hiện tại, mã trạng thái status trả về **200 OK** nếu request thành công.



Hình 10. Chức năng thông báo thùng rác đầy.

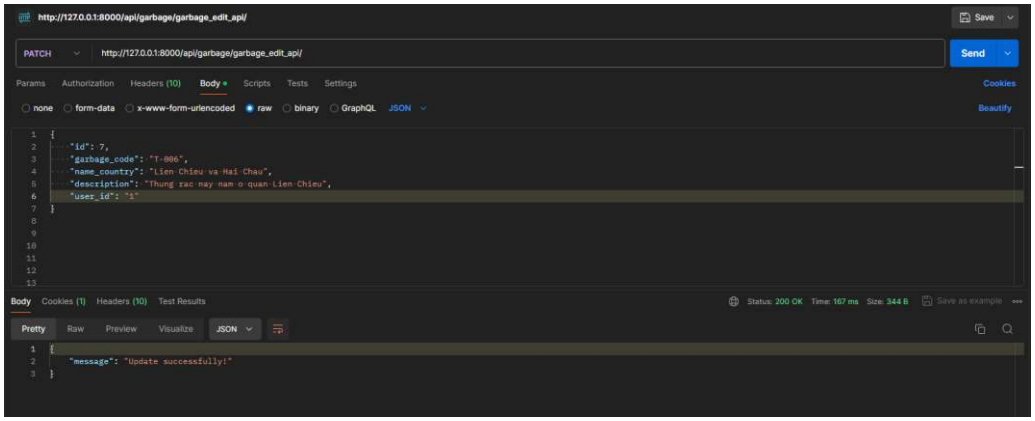
- **Chức năng thêm mới thùng rác:** Người dùng sẽ đăng nhập và có thể thêm mới thùng rác bằng giao thức HTTP POST với dữ liệu đầu vào bao gồm garbage_code(mã thùng rác), name_country(tên khu vực), description(mô tả) và user_id(id của người dùng). Response trả về là một message thông báo thêm thành công và mã trạng thái status là **201 Created.**



Hình 11. Chức năng thêm mới thùng rác.

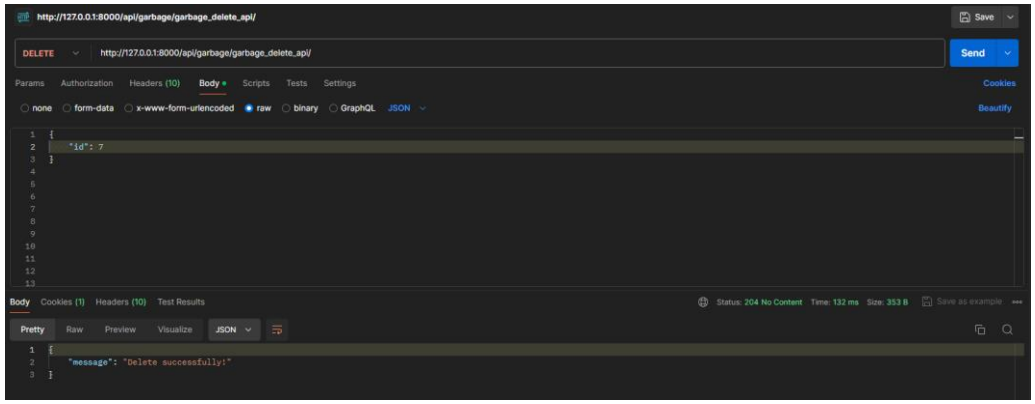
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Chức năng chỉnh sửa thùng rác:** Người dùng sẽ đăng nhập và có thể chỉnh sửa thông tin dùng rác bằng giao thức HTTP PATCH với dữ liệu đầu vào bao gồm id(mã thùng rác), garbage_code(mã thùng rác), name_country(tên khu vực), description(mô tả), user_id(id người dùng). Response trả về là một message thông báo cập nhật thành công và mã trạng thái status là **200 OK.**



Hình 12. Chức năng chỉnh sửa thùng rác.

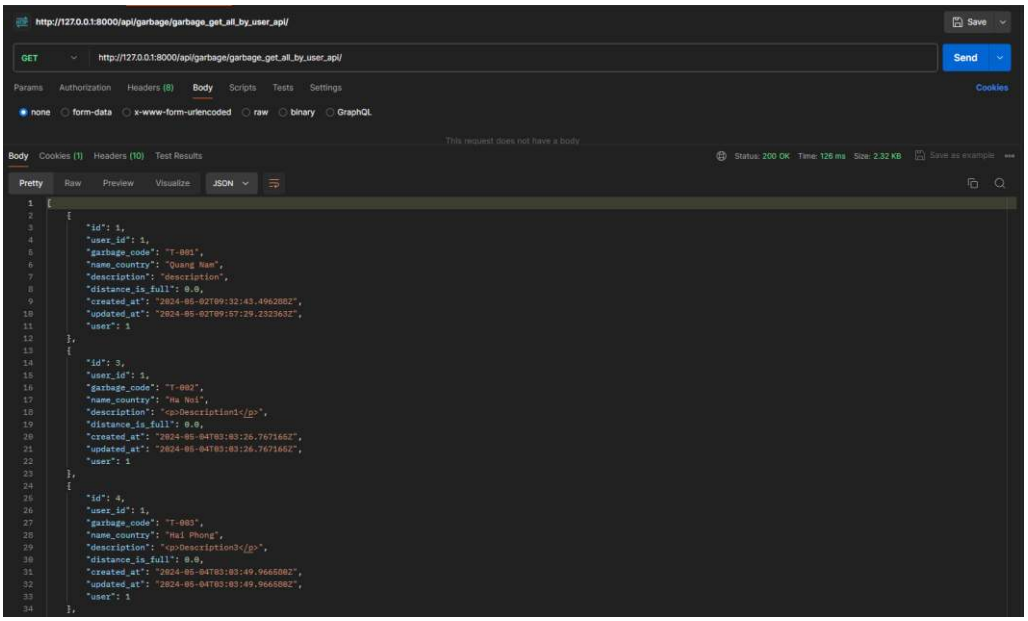
- **Chức năng xoá thùng rác**: Người dùng sẽ đăng nhập và có thể xoá thùng rác mà mình không muốn quản lý bằng giao thức HTTP DELETE với dữ liệu đầu vào bao gồm là id(mã thùng rác) muốn xoá. Response trả về một message thông báo xoá thành công và mã trạng thái status là **204 No Content.**



Hình 13. Chức năng xoá thùng rác.

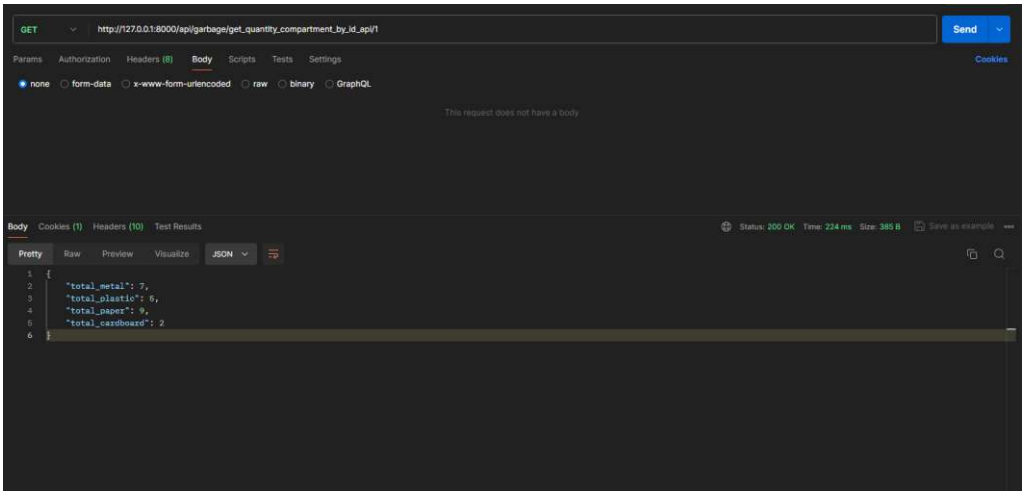
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Chức năng xem tất cả thùng rác bởi một User**: Người dùng có thể xem danh sách tất cả các thùng mà mình quản lý bằng giao thức HTTP GET.Response trả về là một danh sách các thùng rác mà người dùng đó quản lý và mã trạng thái status là **200 OK**



Hình 14. Chức năng xem tất cả thùng rác bởi một User.

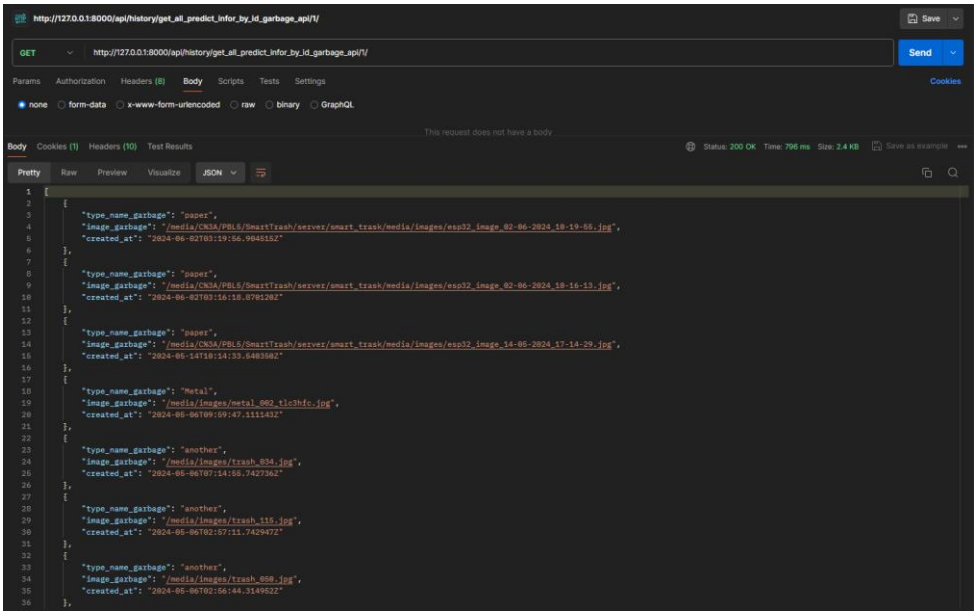
- **Chức năng xem số lượng rác của mỗi ngăn rác:** Người dùng có thể xem số lượng rác của mỗi ngăn của một thùng rác bất kì bằng giao thức HTTP GET với tham số đầu vào là **id** của thùng rác. Response trả về danh sách số lượng rác của mỗi ngăn và mã trạng thái status là **200 OK.**



Hình 15. Chức năng xem số lượng rác của mỗi ngăn rác.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Chức năng xem lịch sử của thùng rác:** Người dùng có thể xem thông tin lịch sử bỏ rác bằng giao giao HTTP GET với tham số đầu vào là **id** của thùng rác. Response trả về danh là một danh sách rác ứng với mỗi loại rác bao gồm tên loại rác, hình ảnh rác và thời gian bỏ rác vào thùng và mã trạng thái status là **200 OK.**



Hình 16. Chức năng xem lịch sử thùng rác.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.3.2. Mobile

- **Bài toán cần giải quyết:**

- Xây dựng app mobile dành cho người quản lý môi trường tương tác với hệ thống gồm những chức năng sau**:**
    - Hiển thị các thùng rác của các khu vực quản lý.
    - Hiển thị số lượng rác thải của mỗi thùng rác: rác thải mỗi loại, loại rác nào quá tải,..
    - Thông báo rác đầy.
    - Dữ liệu rác có trong mỗi thùng rác: hình ảnh rác, ngày giờ, phân loại
    - Điều khiển đóng mở thùng rác.
    - Xem thống kê lượng rác hiện tại trên hệ thống của mỗi khu vực
    - Xem thông tin cá nhân, thông tin của các thùng rác đang quản lý
    - Đổi mật khẩu đăng nhập app
    - Đăng ký, đăng nhập, đăng xuất người dùng

- **Công nghệ sử dụng:**

- Ngôn ngữ lập trình và Framework:
    - **Kotlin**: Ngôn ngữ chính để phát triển ứng dụng Android, nổi bật với sự hiện đại, an toàn và khả năng tương thích ngược với Java.
    - **Android SDK**: Bộ công cụ phát triển phần mềm chính thức của Google cho Android.
- Android UI Framework (UI/UX):
    - **XML Layout**: Sử dụng XML để định nghĩa cấu trúc và giao diện của các Activity, Fragment, View,... Các tệp XML chứa các phần tử UI như LinearLayout, RelativeLayout, TextView, Button, EditText, v.v.
    - **View Binding**: Một tính năng được giới thiệu trong Android để cải thiện và thay thế findViewById(). Tự động tạo các lớp liên kết với các phần tử UI trong layout XML, giúp giảm thiểu mã boilerplate và tăng độ an toàn khi truy cập các phần tử UI.
    - **Jetpack Compos**: Bộ công cụ hiện đại của Android dành cho việc thiết kế giao diện người dùng (UI), giúp tạo ra các UI mượt mà, linh hoạt và dễ bảo trì.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Material Design**: Nguyên tắc thiết kế của Google giúp ứng dụng có giao diện nhất quán, thân thiện với người dùng.

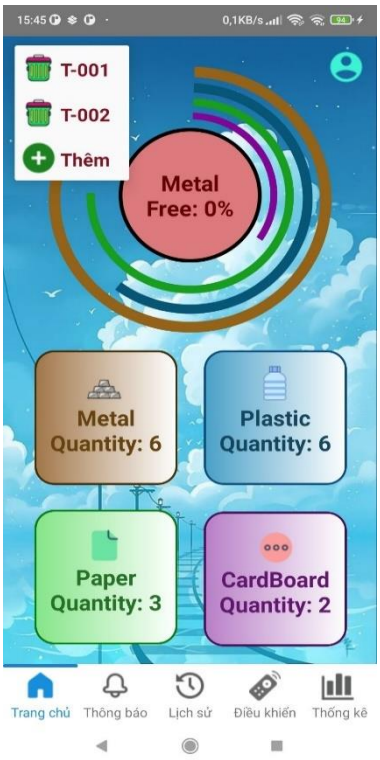
- **Retrofit**: Thư viện chính được sử dụng để tạo các yêu cầu HTTP và xử lý phản hồi từ Sever qua API.
- **GsonConverterFactory**: Trình chuyển đổi được sử dụng để chuyển đổi JSON từ phản hồi của API thành các đối tượng Java hoặc Kotlin sử dụng Gson.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Mockup**

Hình 17. Chức năng đăng kí.

Hình 18. Chức năng đăng nhập.



Hình 20. Danh sách các thùng rác.

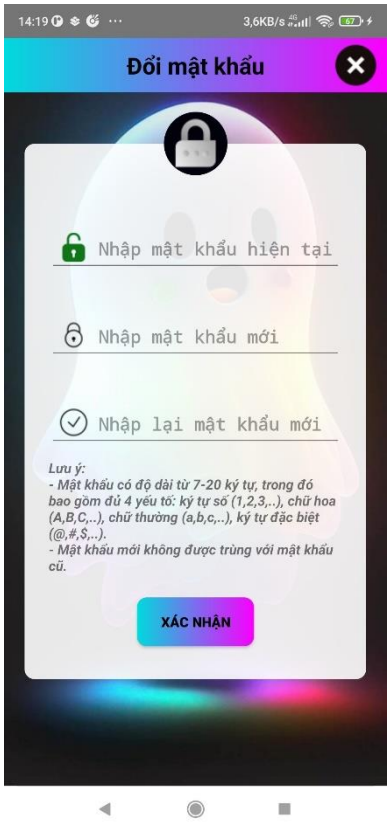
Hình 19. Giao diện trang chủ.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Hình 21. Giao diện tuỳ chọn của User.

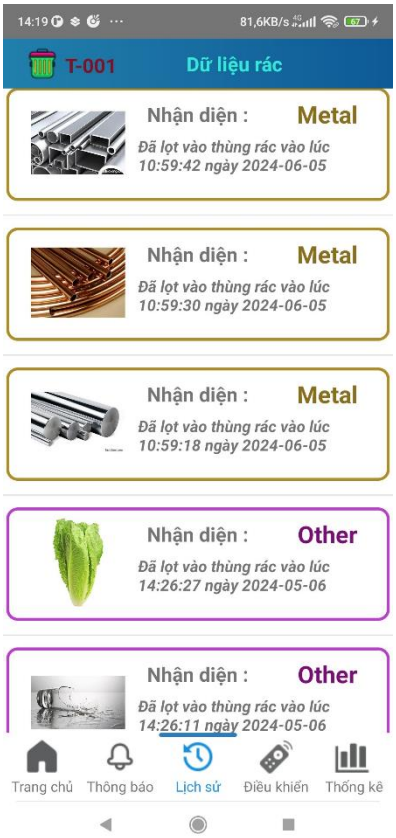
Hình 22. Giao diện thông tin User và thùng rác.

Hình 23. Giao diện thông báo.



Hình 24. Giao diện đổi mật khẩu.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)



Hình 26. Giao diện lịch sử thùng rác.

Hình 25. Giao diện điều khiển thùng rác.

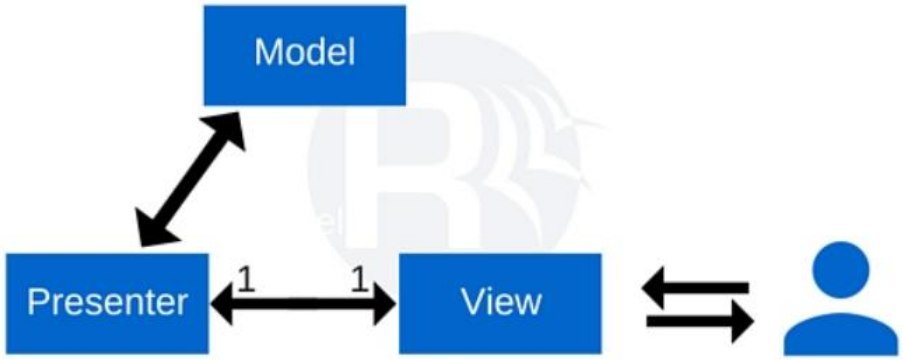
Hình 28. Giao diện thống kê thùng rác.

Hình 27. Giao diện thêm mới thùng rác.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Mô hình kiến trúc MVP (Model-View-Presenter)**

- **Giới thiệu mô hình**



Hình 29. Mô hình MVP.

MVP (Model-View-Presenter) là một mô hình kiến trúc được sử dụng phổ biến trong phát triển phần mềm, đặc biệt là trong các ứng dụng giao diện người dùng (UI). Mô hình này giúp tách biệt rõ ràng các thành phần của ứng dụng, từ đó giúp cho việc quản lý, bảo trì và mở rộng ứng dụng trở nên dễ dàng hơn. MVP có ba thành phần chính:

- Model: Quản lý dữ liệu và logic nghiệp vụ liên quan đến dữ liệu.
- View: Quản lý giao diện người dùng và hiển thị dữ liệu.
- Presenter: Trung gian giữa Model và View, chịu trách nhiệm xử lý logic và tương tác giữa hai thành phần này.

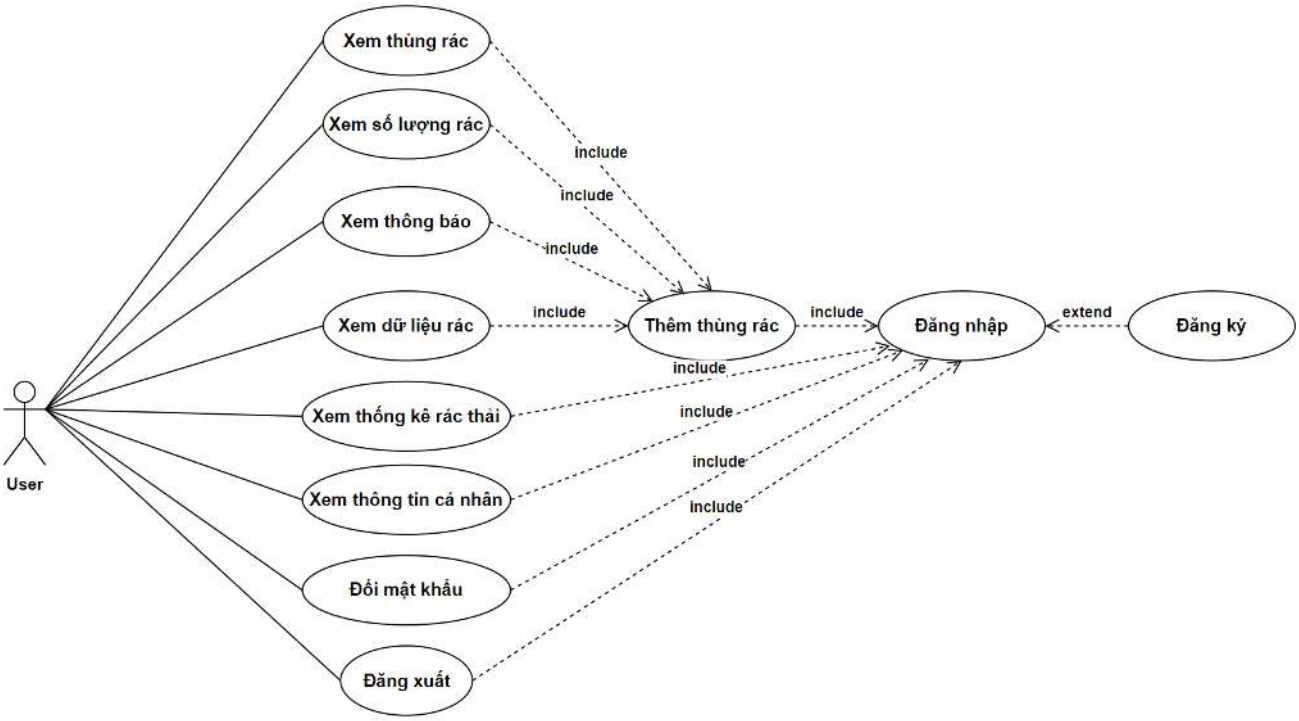
- **Các thành phần trong MVP**
  - **Model**
    - Nhiệm vụ: Quản lý dữ liệu của ứng dụng. Điều này bao gồm việc lấy dữ liệu từ các nguồn khác nhau (API, cơ sở dữ liệu, v.v.), xử lý và lưu trữ dữ liệu (nếu cần).
    - Ví dụ: Trong ứng dụng quản lý rác thải, Model có thể tương tác với một API để lấy danh sách các thùng rác mà người dùng quản lý.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **View**
  - Nhiệm vụ: Hiển thị dữ liệu và quản lý giao diện người dùng. View lắng nghe các sự kiện từ người dùng và chuyển chúng đến Presenter.
  - Ví dụ: Một Activity hoặc Fragment trong ứng dụng Android hiển thị số lượng rác thải của thùng rác mà người dùng quản lý và cung cấp các nút bấm để người dùng tương tác.
- **Presenter**
  - Nhiệm vụ: Trung gian giữa Model và View. Presenter nhận dữ liệu từ Model và chuẩn bị dữ liệu để hiển thị trong View. Presenter cũng nhận các sự kiện từ View và yêu cầu Model thực hiện các thao tác cần thiết.
  - Ví dụ: Presenter sẽ lấy dữ liệu người dùng từ Model khi View khởi tạo và truyền dữ liệu này đến View để hiển thị.
- **Cách thức hoạt động**
  - View nhận sự kiện từ người dùng (như nhấn nút hoặc mở ứng dụng).
  - View chuyển sự kiện này đến Presenter.
  - Presenter xử lý sự kiện và yêu cầu Model lấy hoặc cập nhật dữ liệu.
  - Model thực hiện các thao tác dữ liệu cần thiết (chẳng hạn gọi API, truy vấn cơ sở dữ liệu) và trả về kết quả cho Presenter.
  - Presenter nhận kết quả từ Model, xử lý dữ liệu nếu cần và cập nhật View.
  - View hiển thị dữ liệu mới hoặc thông báo kết quả cho người dùng.

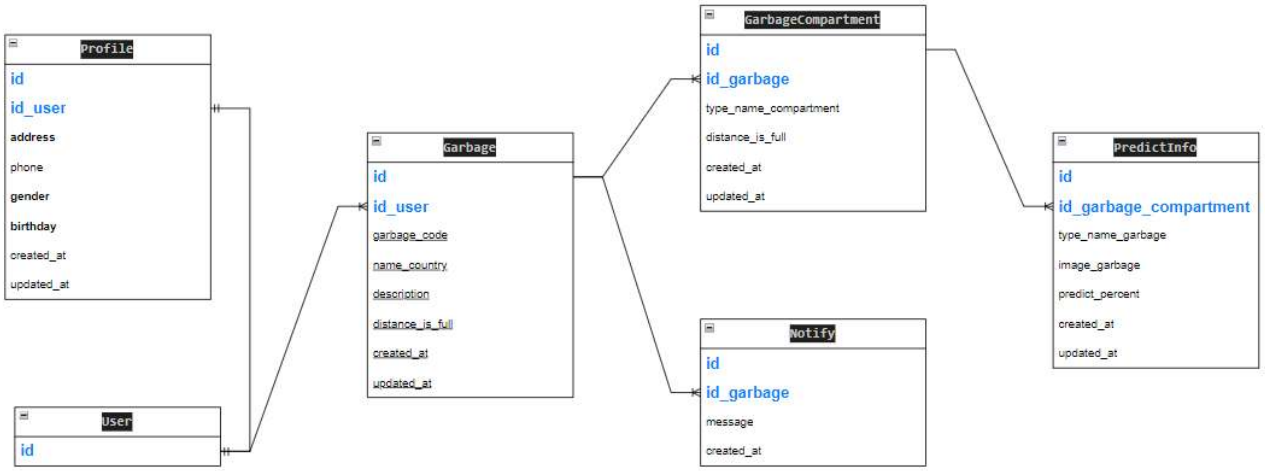
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Sơ đồ usecase diagram:**



Hình 30. Sơ đồ Usecase của hệ thống.

- **Sơ đồ lớp**



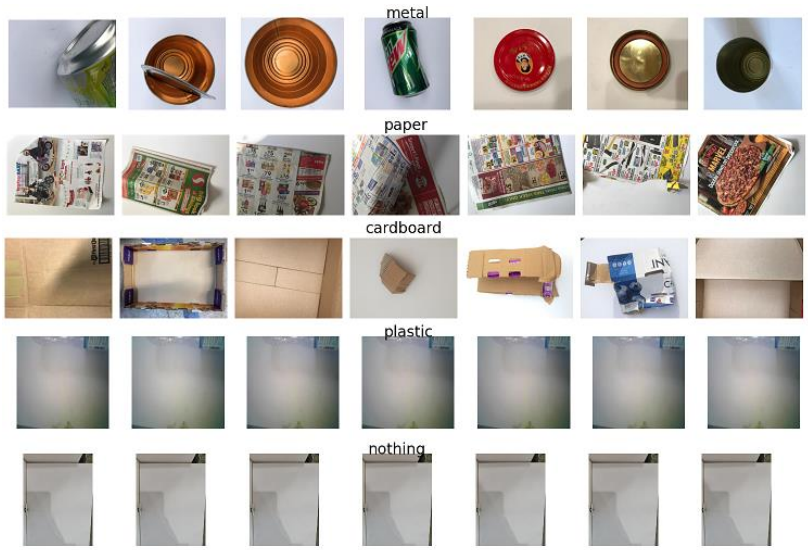
Hình 31. Sơ đồ lớp của hệ thống

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.4. Giải pháp TTNT/KHDL

3.4.1. Thu thập dữ liệu

- Nhóm vừa thu thập dữ liệu ở trên Kaggle và vừa thu thập trực tiếp từ cam ESP32 bao gồm các loại sau:
    - Metal: 461 ảnh
    - Paper: 511 ảnh
    - Cardboard: 537 ảnh
    - Plastic: 482 ảnh
    - Nothing: 400 ảnh
- Dưới đây là các dữ liệu mẫu của từng loại:



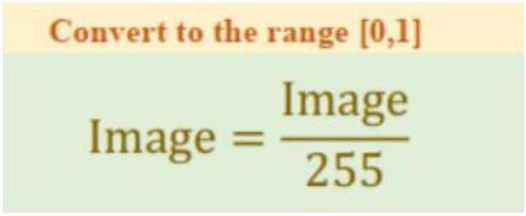
Hình 32. Hình ảnh các mẫu dữ liệu của từng loại rác.

3.4.2. Xử lý dữ liệu.

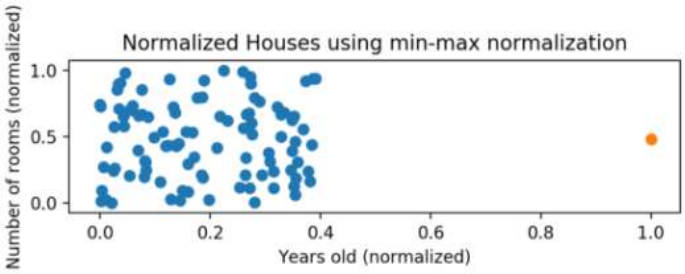
- **Data Normalization**

- Chuẩn hóa dữ liệu là bước quan trọng để cải thiện hiệu suất của các mô hình học máy, giúp mô hình học tốt hơn và tăng tốc độ hội tụ của các thuật toán tối ưu.Khi dữ liệu được đưa về khoảng [0:1] giúp cải thiện hiệu suất của mô hình bằng cách đảm bảo rằng tất cả các đặc trưng có cùng quy mô và do đó không gây ra ảnh hưởng không cân đối đến quá trình huấn luyện.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)



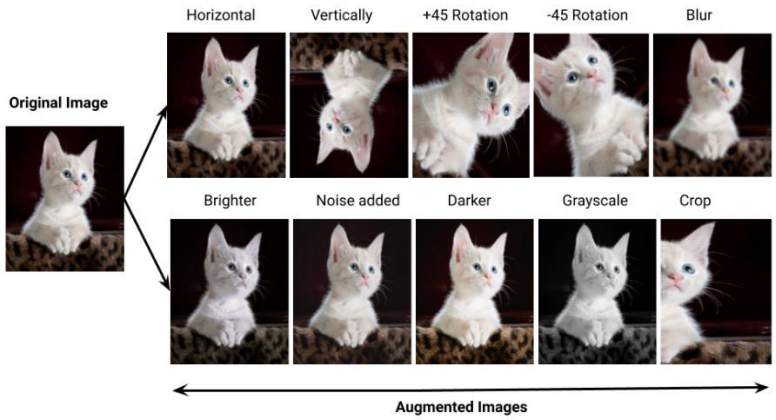
Hình 33. Công thức chuẩn hoá Data Normalizaion.



Hình 34. Phân bố dữ liệu sau khi chuẩn hoá.

- **Data Augmentation**

- Data Augmentation (tăng cường dữ liệu) là một kỹ thuật được sử dụng trong học máy, đặc biệt là trong các bài toán liên quan đến xử lý hình ảnh và ngôn ngữ tự nhiên, nhằm tăng kích thước và độ phong phú của bộ dữ liệu huấn luyện.
- Điều này giúp cải thiện khả năng tổng quát hóa của mô hình bằng cách tạo ra nhiều biến thể khác nhau của dữ liệu gốc mà vẫn giữ nguyên nhãn hoặc ý nghĩa ban đầu.
- Một số phương pháp phổ biến của Augmentation dữ liệu như: lật hình, xoay hình, dịch chuyển, phóng to, thu nhỏ…



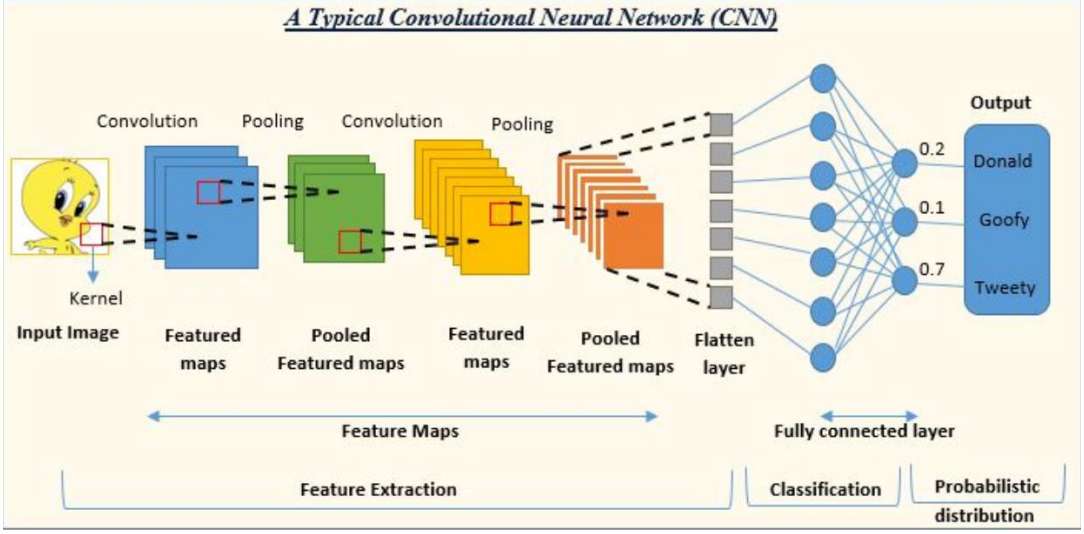
Hình 35. Dữ liệu sau khi Augmentation.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

3.4.3. Xây dựng mô hình AI.

- **Convolutional Neural Netwok**

- Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là CNN là nền tảng siêu siêu quan trọng trong hầu hết các nhiệm vụ trong lĩnh vực computer vision trong một thời gian dài. Đã có rất nhiều (siêu nhiều) những cải tiến liên tiếp. Nhắc tới CNN Architecture, ta có thể list ra một list dài dằng dặc: LeNet, AlexNet, VGG, GoogLeNet, ResNet...Mỗi CNN Architecture đều có thể mạnh của riêng nó, được điều chỉnh để phù hợp với các mục đích khác nhau.
- Với sự phát triển của sự hỗ trợ về phân cứng, GPU, TPU, thế là ngày ngày CNN của chúng ta càng sâu, càng nhiều tham số, càng phức tạp, để tăng độ chính xác lên cao hơn. Đã có những mô hình khổng lồ từ vài trăm triệu param đến số tỷ param.

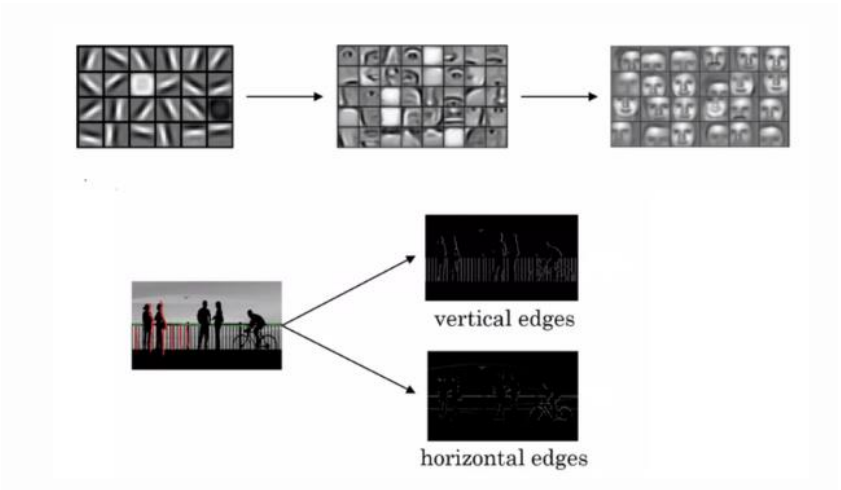


Hình 36. Cấu trúc cơ bản của Convolution Neural Network.

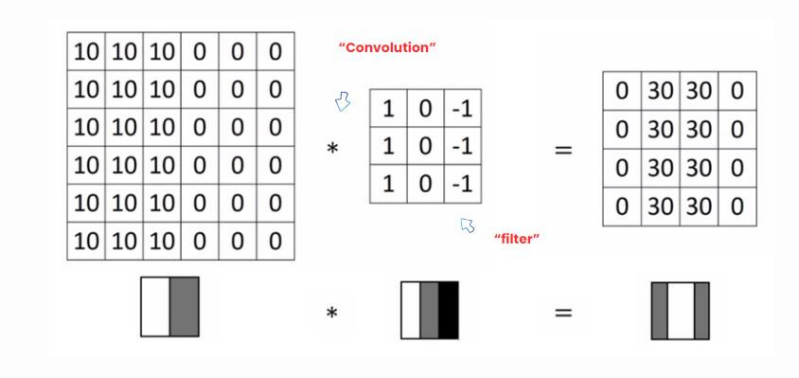
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Convolution**

- Trong mạng nơ-ron sâu, convolution được sử dụng để áp dụng các bộ lọc (filters) lên dữ liệu đầu vào để tạo ra các biểu diễn mới có chứa thông tin về các đặc trưng cụ thể của dữ liệu, như cạnh, góc, màu sắc, và các đặc điểm nổi bật khác. Quá trình convolution này giúp mạng nơ-ron tự động học các đặc trưng cần thiết để thực hiện các tác vụ như nhận dạng hình ảnh hoặc phân loại.

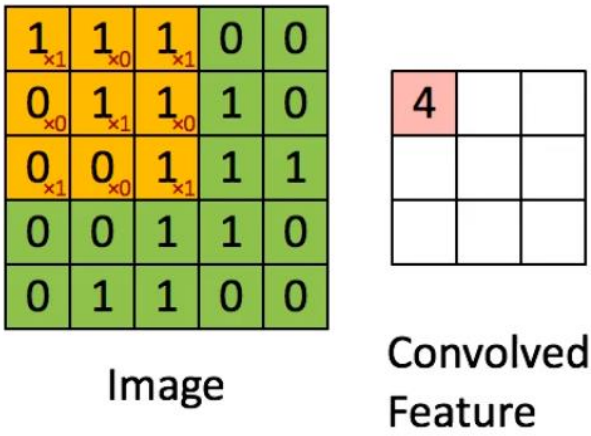


Hình 37. Sử dụng CNN trong bài toán phát hiện cạnh.



Hình 38. Tích chập trong bài toán phát hiện cạnh.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

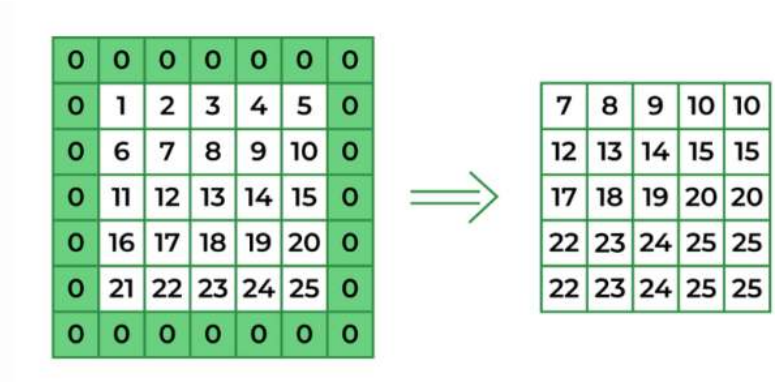


Hình 39. Trực quan phép tích chập.

- **Padding Convolution**

- Padding là một kỹ thuật được sử dụng trong quá trình convolution trong mạng nơ-ron sâu để điều chỉnh kích thước của đầu vào trước khi áp dụng bộ lọc (filter) lên nó. Padding thêm các giá trị (thường là 0) vào các cạnh của đầu vào để tạo ra một lớp đầu vào mở rộng, từ đó giữ nguyên kích thước của đầu ra sau khi quá trình convolution được thực hiện.

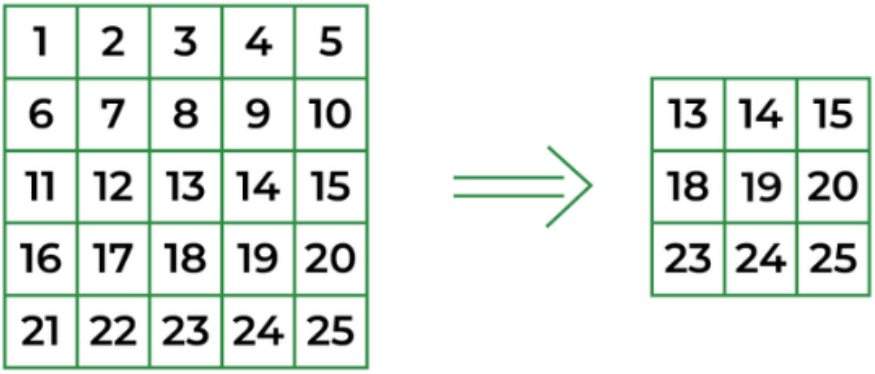
- **Same Padding (Padding cùng kích thước):** Kỹ thuật này sẽ thêm số lượng giá trị padding sao cho đầu ra của quá trình convolution có kích thước giống với đầu vào ban đầu. Điều này giúp đảm bảo rằng không có sự thay đổi về kích thước của đầu vào khi thực hiện convolution.



Hình 40. Same Padding Convolution.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

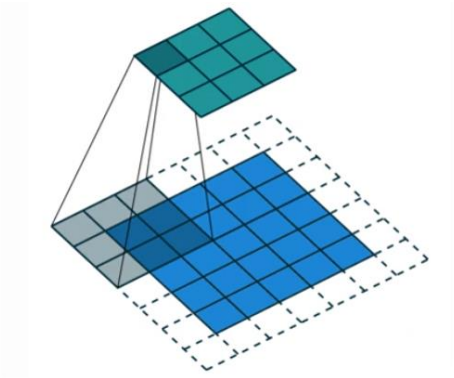
- **Valid Padding (Padding không)**: Trong trường hợp này, không có padding được thêm vào, do đó kích thước của đầu ra sẽ nhỏ hơn so với kích thước của đầu vào.



Hình 41. Valid Padding Convolution.

- **Strided Convolution**

- Strided Convolution là một phương pháp trong quá trình convolution trong mạng nơ-ron sâu, nơi mà bước di chuyển của bộ lọc (filter) được áp dụng trên dữ liệu đầu vào không phải là 1, mà là một bước nhảy được xác định trước, được gọi là stride
- Strided Convolution thường được sử dụng để giảm kích thước của đầu ra, giảm chi phí tính toán và tăng tốc độ huấn luyện trong quá trình xử lý ảnh và video trong các mạng nơ-ron sâu.Tuy nhiên, việc sử dụng stride lớn có thể dẫn đến việc mất mát thông tin trong quá trình convolution, đặc biệt là ở các vùng biên của ảnh.

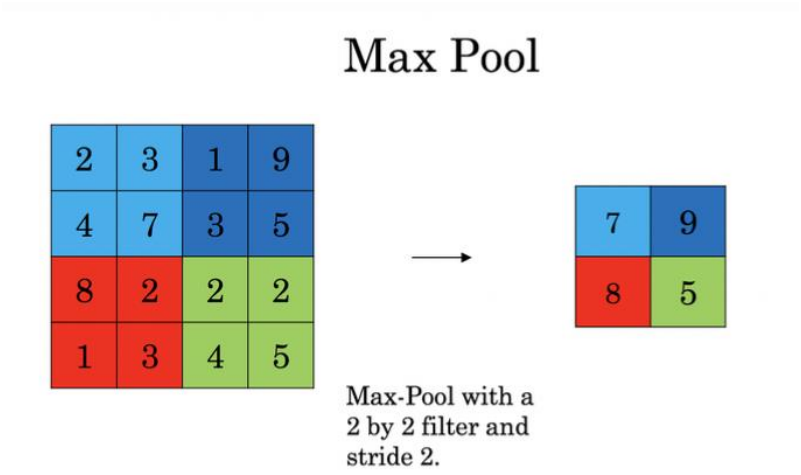


Hình 42. Stride Convolution.

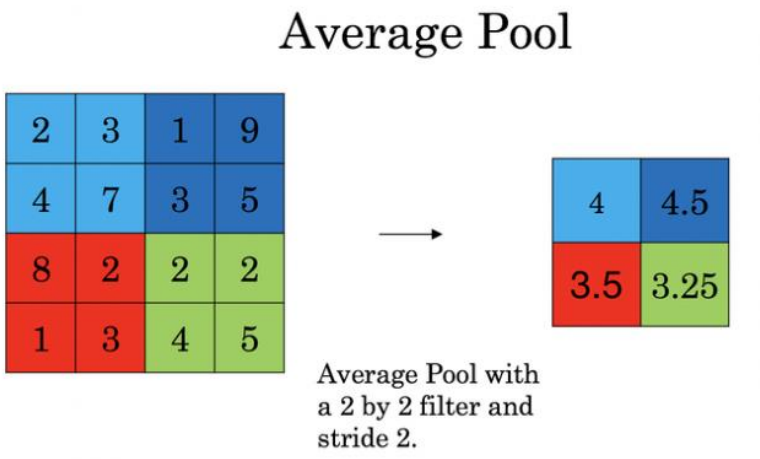
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Pooling Layer**

- Pooling Layer là một lớp quan trọng trong mạng nơ-ron sâu, được sử dụng để giảm kích thước không gian của biểu diễn đặc trưng, giúp giảm chi phí tính toán và kiểm soát overfitting trong quá trình huấn luyện.
- Tại mỗi vị trí của cửa sổ (filter), một phép toán được áp dụng để tính toán giá trị đạidiện cho các vùng dữ liệu. Các phép toán phổ biến trong Pooling Layer là Max Pooling (chọn giá trị lớn nhất) hoặc Average Pooling (tính trung bình)



Hình 43. Max Pooling.

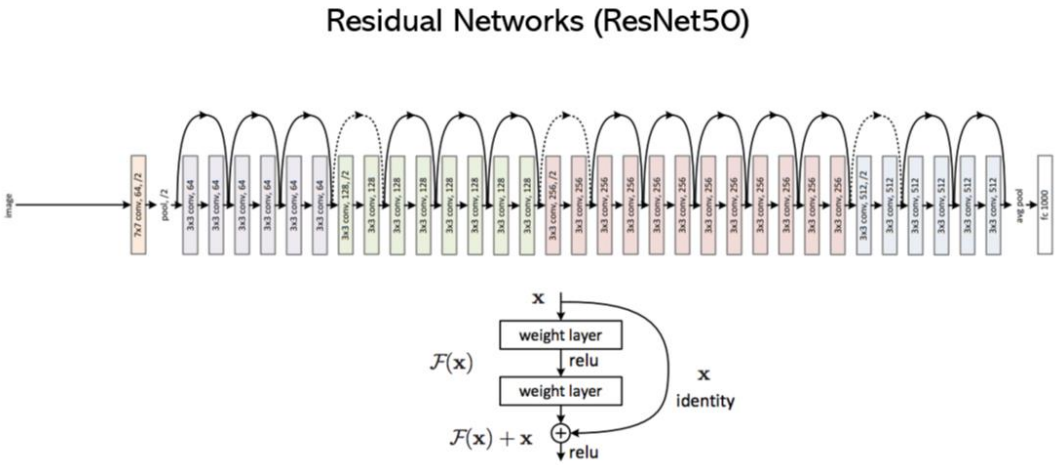


Hình 44. Average Pooling

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Kiến trúc mạng Resnet50**

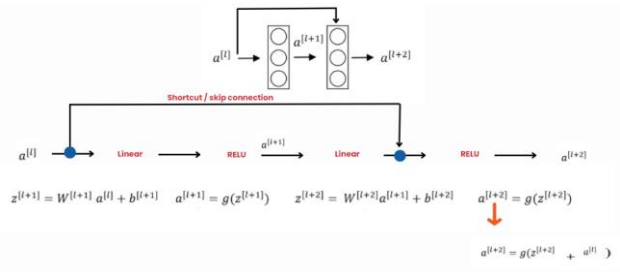
- Để thực hiện việc huấn luyện mô hình, chúng em sẽ sử dụng kiến trúc mạng Resnet50 (Residual Network), đây là một kiến trúc mạng nơ-ron tích chập CNN(Convolutional Neural Network) được thiết kế để giải quyết vấn đề khi huấn luyện các mạng rất sâu. Resnet50 được sử dụng rộng rãi trong các ứng dụng thị giác máy tính như phân loại hình ảnh, phát hiện đối tượng, phân đoạn. Mạng này được giới thiệu bởi Kaiming He và các đồng nghiệp trong bài báo "Deep Residual Learning for Image Recognition" năm 2015, và đã giành giải thưởng Best Paper tại hội nghị CVPR 2016.



Hình 45. Kiến trúc mạng ResNet50

- **Khối Residual Connection**

- Điểm mới lạ chính của ResNet là việc giới thiệu các "khối dư" (residual blocks). Thay vì học một ánh xạ trực tiếp từ đầu vào đến đầu ra, các khối này học phần dư (difference) giữa đầu vào và đầu ra. Điều này đạt được thông qua việc thêm các kết nối tắt (shortcut connections) vượt qua một hoặc nhiều lớp.

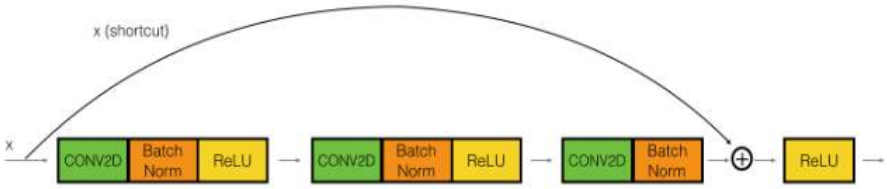


Hình 46. Skip Connection.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Quá trình thực hiện mô hình**

❖ **Xây dựng Identity Block**



Hình 47. Khối Identity Block.

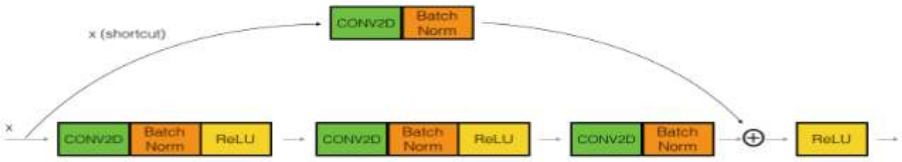
- Đây là một khối được thiết kế để duy trì kích thước của đầu vào và đầu ra. Khối này giúp giảm thiểu vấn đề gradient biến mất trong mạng nơ-ron sâu. Dưới đây là các bước xây dựng:
  - Input: Đầu vào của khối nhận dạng, giả sử là $a[l]$
  - Convolutional Layer (CONV2D):
    - Áp dụng một lớp tích chập (Convolutional Layer) để trích xuất các đặc trưng.
    - Kích thước bộ lọc và số lượng bộ lọc được xác định trước.
    - Thêm hàm kích hoạt ReLU sau lớp tích chập để giới thiệu tính phi tuyến.
  - Batch Normalization (BatchNorm):
    - Áp dụng Batch Normalization sau mỗi lớp tích chập để chuẩn hóa đầu ra, giúp tăng tốc độ huấn luyện và ổn định mạng.
  - Convolutional Layer (CONV2D):
    - Áp dụng lớp tích chập tiếp theo trên đầu ra từ bước trước.
    - Thêm hàm kích hoạt ReLU sau lớp tích chập này.
  - Batch Normalization (BatchNorm):
    - Áp dụng Batch Normalization sau lớp tích chập thứ hai.
  - Convolutional Layer (CONV2D):
    - Áp dụng lớp tích chập thứ ba trên đầu ra từ bước trước.
    - Kích thước bộ lọc và số lượng bộ lọc được xác định trước.
  - Batch Normalization (BatchNorm):
    - Áp dụng Batch Normalization sau lớp tích chập thứ ba.
  - Addition (Add):

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Thực hiện phép cộng giữa đầu vào ban đầu và đầu ra từ bước trước (sau khi đã qua ba lớp tích chập và BatchNorm).
- Đây là bước "kết nối tắt" (shortcut connection) giúp thông tin truyền qua mạng một cách hiệu quả hơn.

- Activation (ReLU):
  - Áp dụng hàm kích hoạt ReLU lên kết quả của phép cộng để tạo đầu ra cuối cùng của khối nhận dạng.

❖ **Xây dựng khối Convolutional Block**



Hình 48. Khối Convolution Block.

- Input:Đầu vào của khối tích chập, giả sử là $a[l]$
- Lưu Trữ Đầu Vào Ban Đầu:
  - Lưu giá trị của đầu vào ban đầu để sử dụng sau này trong bước cộng.
- Thành Phần Đầu Tiên của Đường Chính:
  - Áp dụng một lớp tích chập (Convolutional Layer) với kích thước bộ lọc là 1x1 và stride là s. Lớp này giúp thay đổi số lượng kênh và giảm kích thước không gian.
  - Áp dụng lớp chuẩn hóa batch (Batch Normalization) để chuẩn hóa đầu ra.
  - Áp dụng hàm kích hoạt (ReLU) để giới thiệu tính phi tuyến.
- Thành Phần Thứ Hai của Đường Chính:
  - Áp dụng một lớp tích chập khác với kích thước bộ lọc là fx f và stride là 1. Lớp này giúp trích xuất các đặc trưng.
  - Áp dụng lớp chuẩn hóa batch và hàm kích hoạt ReLU.

- Thành Phần Thứ Ba của Đường Chính:
  - Áp dụng lớp tích chập thứ ba với kích thước bộ lọc là 1x1 và stride là 1. Lớp này giúp khôi phục lại số lượng kênh ban đầu.

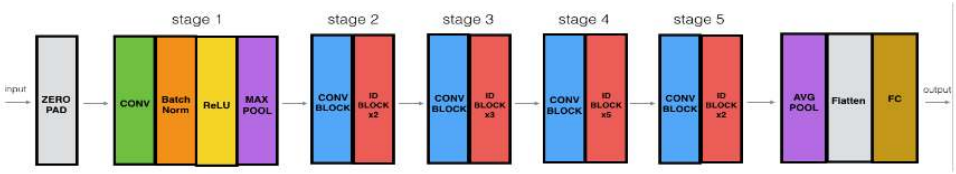
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Áp dụng lớp chuẩn hóa batch để chuẩn hóa đầu ra.

- Đường Tắt (Shortcut Path):
  - Áp dụng một lớp tích chập với kích thước bộ lọc là 1x1 và stride là s trên đầu vào ban đầu. Lớp này giúp điều chỉnh kích thước của đầu vào để phù hợp với đầu ra của đường chính.
  - Áp dụng lớp chuẩn hóa batch để chuẩn hóa đầu ra của đường tắt.
- Kết Hợp Đường Chính và Đường Tắt:
  - Thực hiện phép cộng giữa đầu ra của đường chính và đầu ra của đường tắt. Đây là bước "kết nối tắt" (shortcut connection) giúp thông tin truyền qua mạng một cách hiệu quả hơn.
- Áp Dụng Hàm Kích Hoạt Cuối Cùng:
  - Áp dụng hàm kích hoạt ReLU lên kết quả của phép cộng để tạo đầu ra cuối cùng của khối tích chập.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

❖ **Kết hợp Identiy Block và Convolutional Block để xây dựng mạng.**



Hình 49. Kết hợp Indentity Block và Convolutional Block.

- Zero-padding thêm lớp đệm có kích thước (3,3) vào đầu vào.
- Stage 1:
  - Phép tích chập 2D có 64 bộ lọc kích thước (7,7) và sử dụng bước nhảy là (2,2).
  - BatchNorm được áp dụng vào trục 'channels' của đầu vào.
  - MaxPooling sử dụng cửa sổ (3,3) và bước nhảy là (2,2).
- Stage 2:
  - Khối **Convolution Block** sử dụng ba bộ lọc kích thước [64,64,256], filter là 3, và stride là 1.
  - Hai khối **Identity Block** sử dụng ba bộ lọc kích thước [64,64,256], và filter là 3.
- Stage 3:
  - Khối **Convolution Block** sử dụng ba bộ lọc kích thước [128,128,512], filter là 3 và stride là 2.
  - Ba khối **Identity Block** sử dụng ba bộ lọc kích thước [128,128,512] và filter là 3.
- Stage 4:
  - Khối **Convolution Block** sử dụng ba bộ lọc kích thước [256, 256, 1024], filter là 3 và stride là 2.
  - Năm khối **Identity Block** sử dụng ba bộ lọc kích thước [256, 256, 1024] và filter là 3.
- Stage 5:
  - Khối **Convolution Block** sử dụng ba bộ lọc kích thước [512, 512, 2048], filter là 3 và stride là 2.

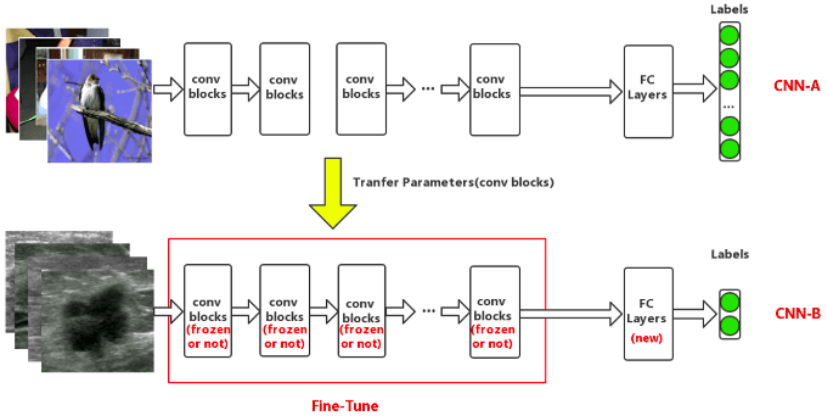
This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Hai khối **Identity Block** sử dụng ba bộ lọc kích thước [512, 512, 2048] và filter là 3.

- MaxPooling 2D sử dụng cửa sổ (pool_size) kích thước (2,2).
- Lớp 'flatten' không có thông số siêu tham số.
- Lớp Fully Connected (Dense) giảm đầu vào của nó xuống số lớp sử dụng hàm kích hoạt softmax.

❖ **Transfer Learning mô hình.**

- Sau khi quan sát và đánh giá được hiệu suất của mô hình nhóm vừa xây dựng từ đầu thì nhận thấy độ chính xác của mô hình chưa cao, mô hình vẫn chưa khái quá hoá tốt trên tập dữ liệu mới. Vì thế nhóm đã chuyển sang huấn luyện mô hình bằng phương pháp Transfer Learning (học chuyển giao)



Hình 50. Ví dụ về Transfer Learing

- Nhóm sẽ tạo một mô hình học chuyển giao sử dụng ResNet50, một mô hình mạng đã được huấn luyện trước trên bộ dữ liệu ImageNet.Dưới đây là bước thực hiện:

  - Khởi tạo mô hình RestNet50 từ thư viện Tensflow với trọng số được được là **'imagenet'**, chúng ta sẽ loại bỏ các lớp đầu ra của mô hình (các lớp Fully Connected), kích thước đầu vào của mô hình là (350,350,3).
  - Chúng ta sẽ đóng băng 12 layer đầu tiên của mô hình bằng cách đặt tham số **trainable = False**, điều này có nghĩa là các lớp này sẽ không được cập nhật trọng số trong quá trình huấn luyện. Giúp giữ nguyên các trọng số đã được huấn luyện trước và chỉ huấn luyện các lớp mới.
  - Thêm các layer mới vào để phù hợp với bài toán mới, nhóm đã thêm các lớp sau:

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **GlobalAveragePooling2D():** Lớp pooling trung bình toàn cục, giúp giảm số lượng tham số và tránh overfitting.
- **Dropout(0.5):** Lớp Dropout với tỷ lệ dropout là 0.5, ngắt ngẫu nhiên 50% số kết nối trong lớp để giảm overfitting.
- **Dense(5, activation='softmax'):** Lớp Dense với 5 đầu ra và hàm kích hoạt softmax, thường được sử dụng cho bài toán phân loại với 5 lớp.

- Cuối cùng chúng ta sẽ có được mô hình mới với đầu vào của mô hình ResNet50 và đầu ra là các lớp tuỳ chỉnh vừa thêm vào.Mô hình này được sử dụng các đặc trưng đã được học của ResNet50 và các lớp đầu ra mới để thực hiện nhiệm vụ phân loại mới.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

CHƯƠNG 3. KẾT QUẢ

3.1 Kết quả nhận diện

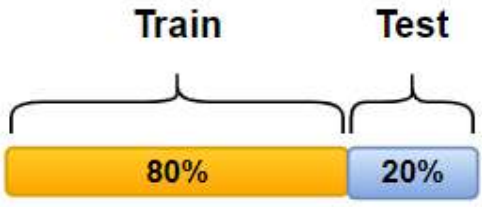
❖ **Dữ liệu đầu vào**

- Độ phân giải của ảnh: 384 x 384 Pixel
- Kích thước trung bình của file ảnh: 30KB

❖ **Dữ liệu đầu ra:** Kết quả phân loại rác

❖ **Dữ liệu sử dụng để huấn luyện mô hình**

- Nguồn gốc: thu thập từ Kaggle và nhóm tự chụp bằng cam ESP32
- Các tính chất của ảnh:
  - Ảnh từ Kaggle: 512 x 384 Pixels ~ 20KB
  - Ảnh tự chụp: 384 x 384 Pixels ~ 10KB
- Sau khi thu thập ảnh xong ta sẽ có kết quả như sau:
  - Metal: 461 ảnh
  - Paper: 511 ảnh
  - Cardboard: 537 ảnh
  - Plastic: 482 ảnh
  - Nothing: 400 ảnh
- Phân chia dữ liệu để huấn luyện mô hình:
  - Train: 80%
  - Test: 20%



Hình 51. Phân chia dữ liệu để huấn luyện mô hình.

❖ **Công cụ sử dụng:** Google Colab (T4-GPU)

❖ **Điều kiện tiến hành thực nghiệm:**

- Bảo đảm điều kiện ánh sáng từ môi trường
- Độ chính xác của hệ thống phụ thuộc lớn vào góc chụp trong quá trình lấy ảnh thông qua camera. Sau khi nhóm đã thực hiện trong điều kiện camera không lấy

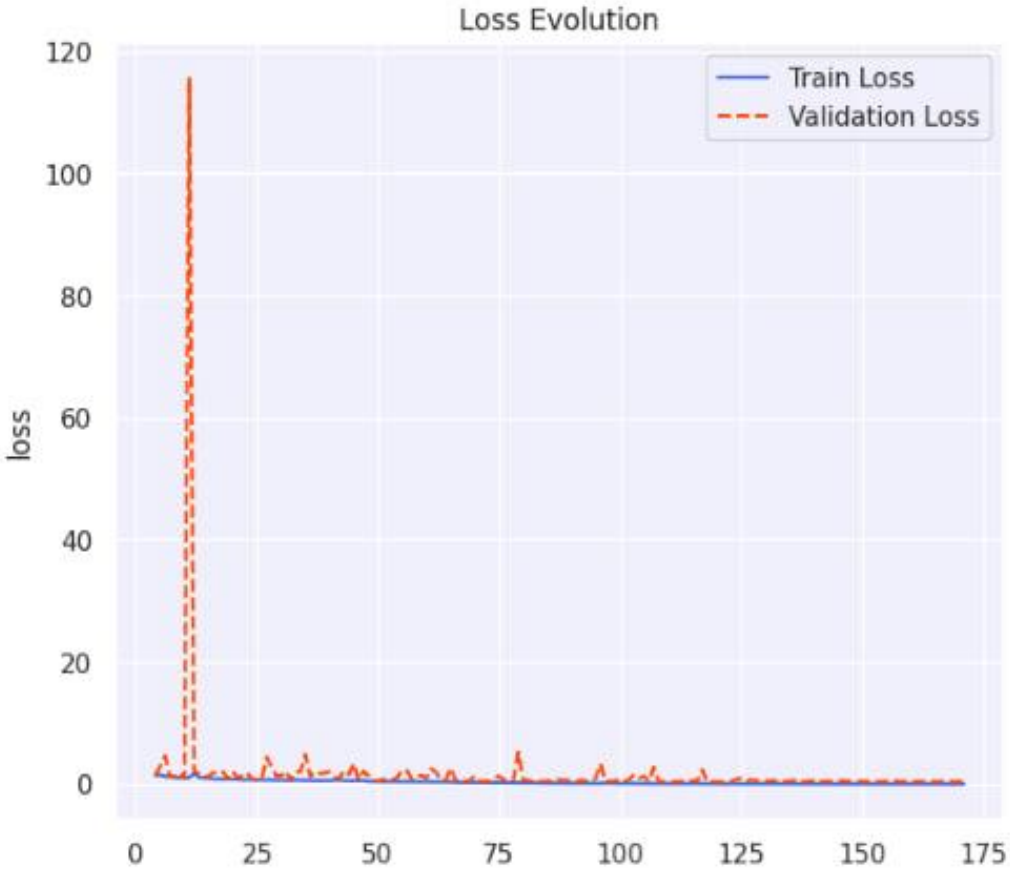
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

hết được hình ảnh thì kết quả trả về sẽ sai sót hơn so với góc chụp lấy được toàn bộ, do đó cần chú ý góc đặt rác.

- Không bỏ nhiều loại rác cùng một lúc.

❖ **Độ chính xác:**

- Độ chính xác trong bài toán được đánh giá dựa trên hàm mất mát (Loss) và độ chính xác (Accuracy)
  - Loss: đánh giá độ sai khác giữa dự đoán đầu ra và thực tế dựa trên hàm Cross-entropy - đây là hàm được sử dụng nhiều trong các bài toán phân loại. Cross-entropy càng nhỏ thì độ chính xác phân loại càng lớn
  - Accuracy: đánh giá hiệu suất mô hình dựa trên tỉ lệ dự đoán đúng / tổng số lượng dự đoán
- Kết quả của mô hình được xây dựng từ đầu:
  - Đồ thị hàm Loss



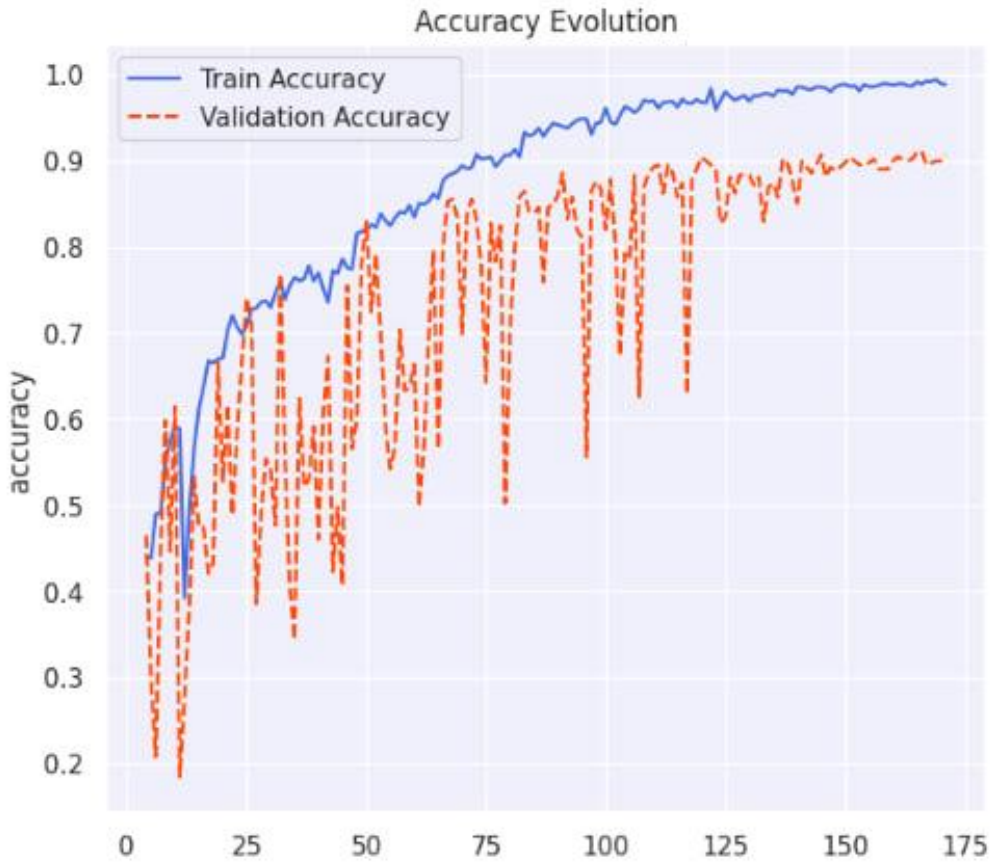
Hình 52. Đồ thị hàm Loss của mô hình xây dựng từ đầu.

- Validation Loss có một đỉnh rất cao ngay từ đầu, có thể là do mô hình đang cố gắng điều chỉnh và chưa tìm ra cách tối ưu hóa tốt. Sau khoảng

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

25 epoch, cả hai loại mất mát duy trì ở mức thấp và gần như không thay đổi mô hình đã hội tụ và không còn học được nhiều từ dữ liệu mới.

- Đồ thị Accuracy

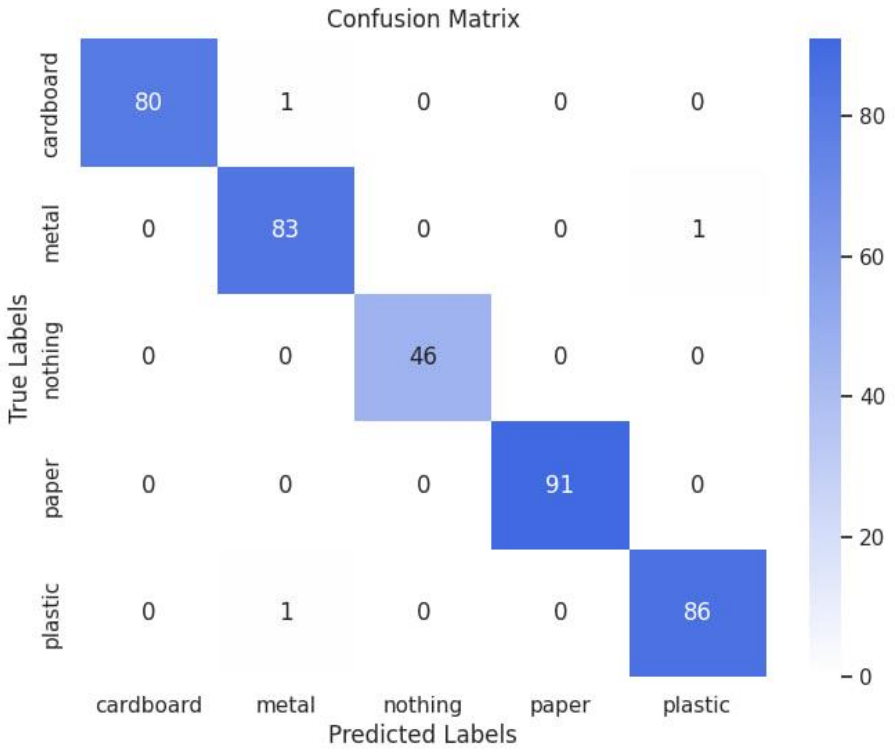


Hình 53. Đồ thị Accuracy của mô hình xây dựng từ đầu.

- Mô hình học tốt từ dữ liệu huấn luyện, và độ chính xác của mô hình trên tập kiểm tra cũng tăng lên theo thời gian, cho thấy mô hình có khả năng tổng quát hóa tốt. Có sự dao động lớn trong Validation Accuracy có thể là dấu hiệu cần xem xét kỹ thuật giảm nhiễu hoặc điều chỉnh hyperparameters để cải thiện sự ổn định của mô hình.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Confusion Matrix



Hình 54. Confusion Matrix của mô hình xây dựng từ đầu.

- Mô hình hoạt động khá tốt với hầu hết các nhãn, với các giá trị nằm chủ yếu trên đường chéo chính của ma trận nhầm lẫn. Điều này cho thấy mô hình phân loại chính xác phần lớn các mẫu dữ liệu.
  - **Cardboard:** Có 80 mẫu "cardboard" được phân loại chính xác. Có 1 mẫu bị phân loại sai thành "metal".
  - **Metal:** Có 83 mẫu "metal" được phân loại chính xác. Có 1 mẫu " bị phân loại sai thành "plastic".
  - **Nothing:**Có 46 mẫu "nothing" được phân loại chính xác.Không có mẫu "nothing" nào bị phân loại sai.
  - **Paper:** Có 91 mẫu "paper" được phân loại chính xác. Không có mẫu "Paper" nào bị phân loại nhầm.
  - **Plastic:**Có 86 mẫu "plastic" được phân loại chính xác.Có 1 mẫu "plastic" bị phân loại sai thành "metal"

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Classification Report

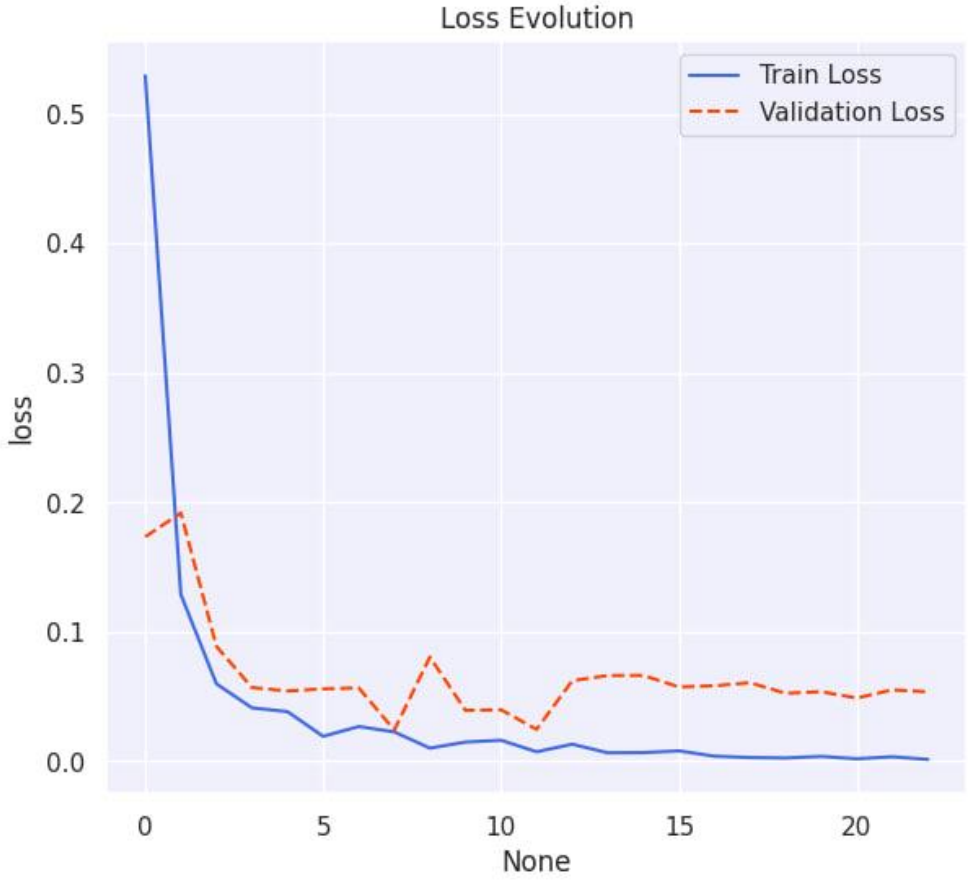
	Precision	Recall	F1-Score	Support
Cardboard	0.92	0.94	0.93	81
Metal	0.90	0.87	0.88	82
Nothing	1.00	1.00	1.00	46
Paper	0.9	0.88	0.89	123
Plastic	0.84	0.88	0.86	87
Accuracy			0.9	389
Macro avg	0.91	0.91	0.91	389
Weighted avg	0.9	0.9	0.9	389

  - Từ đồ thị chúng ta thấy độ chính xác trên tập huấn luyện và trên tập kiểm tra rất cao, 92% với tập huấn luyện và 90% đối với tập kiểm tra. Điều đó cho thấy mô hình không bị Overfiting, hay Underfiting.Mô hình đã khái quát hoá tốt trên tập dữ liệu mới.

- Tốc độ thực thi: 1.8 ~ 1.9

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Kết quả của mô hình sử dụng Transfer Learning:
    - **Đồ thị hàm Loss**

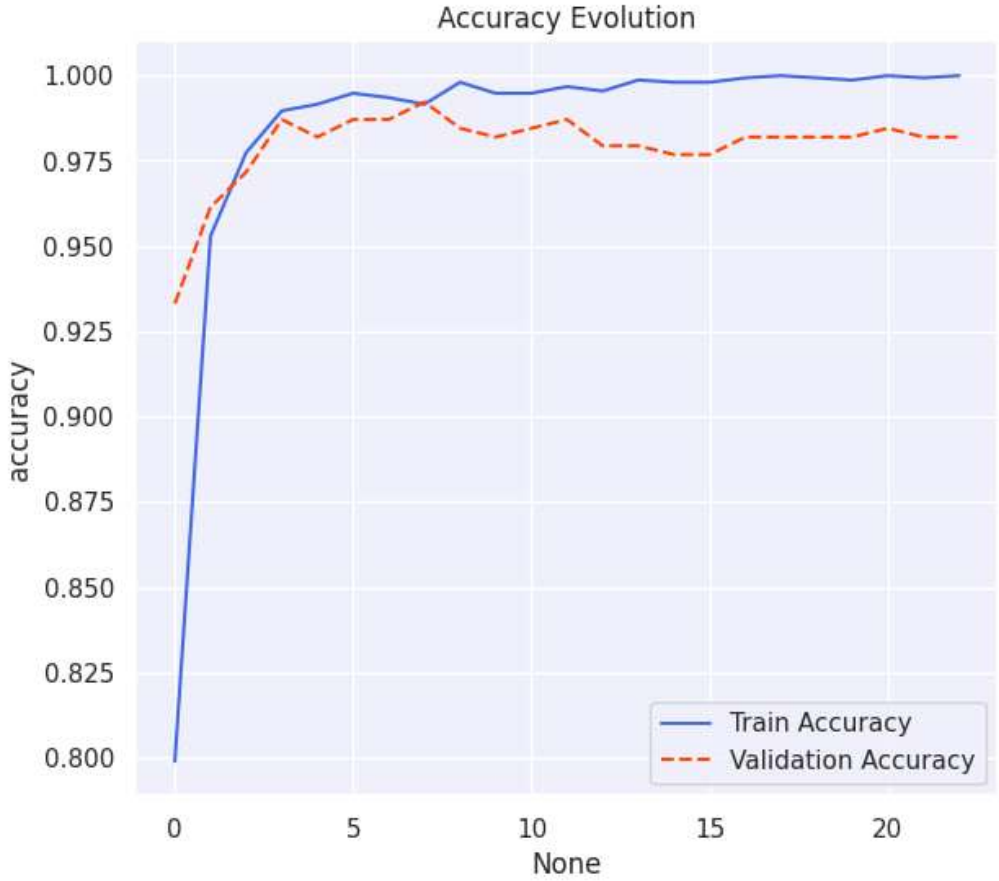


Hình 55. Đồ thị hàm Loss của mô hình Transfer Learning.

- Độ sai khác giữa tập huấn luyện và tập kiểm tra có khoảng cách khá nhỏ, điều đó cho thấy mô hình đã học được tốt trên tập dữ liệu huấn luyện và khái quát hoá tốt dữ liệu trên tập kiểm tra

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Đồ thị Accuracy**

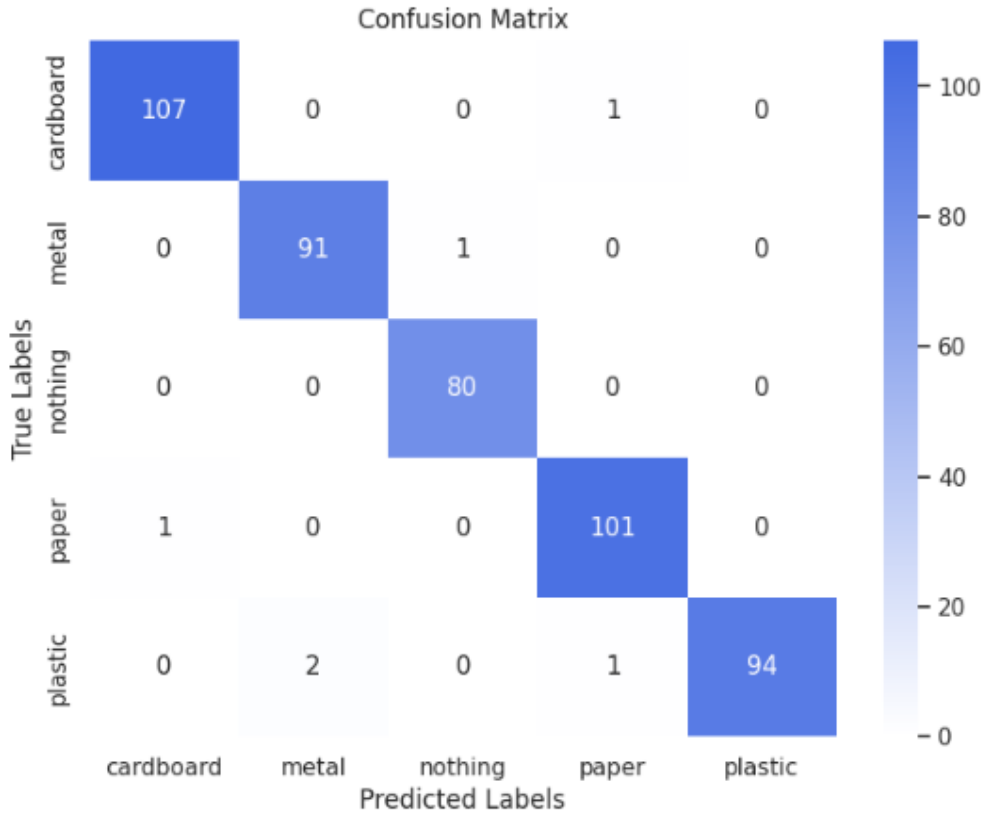


Hình 56. Đồ thị Accuracy của mô hình Transfer Learning.

- Từ đồ thị chúng ta thấy độ chính xác trên tập huấn luyện và trên tập kiểm tra rất cao, 99% với tập huấn luyện và 97% đối với tập kiểm tra. Điều đó cho thấy mô hình không bị Overfiting, hay Underfiting.Mô hình đã khái quát hoá tốt trên tập dữ liệu mới.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Confusion Matrix:**



Hình 57. Confusion Matrix của mô hình sử dụng Transfer Learning.

- Mô hình hoạt động khá tốt với hầu hết các nhãn, với các giá trị nằm chủ yếu trên đường chéo chính của ma trận nhầm lẫn. Điều này cho thấy mô hình phân loại chính xác phần lớn các mẫu dữ liệu.
    - **Cardboard:** Có 107 mẫu "cardboard" được phân loại chính xác. Có 1 mẫu "cardboard" bị phân loại sai thành "paper".
    - **Metal:** Có 91 mẫu "metal" được phân loại chính xác. Có 1 mẫu "metal" bị phân loại sai thành "nothing".
    - **Nothing:**Có 80 mẫu "nothing" được phân loại chính xác.Không có mẫu "nothing" nào bị phân loại sai.
    - **Paper:** Có 101 mẫu "paper" được phân loại chính xác. Có 1 mẫu "paper" nào bị phân loại sai thành “cardboard”.
    - **Plastic:**Có 94 mẫu "plastic" được phân loại chính xác.Có 1 mẫu "plastic" bị phân loại sai thành "paper", 2 mẫu bị phân loại sai thành “metal”.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- **Classification Report**

	Precision	Recall	F1-Score	Support
Cardboard	0.99	0.99	0.99	108
Metal	0.98	0.99	0.98	92
Nothing	0.99	1.00	0.99	80
Paper	0.98	0.99	0.99	102
Plastic	1.00	0.97	0.98	97
Accuracy			0.99	497
Macro avg	0.99	0.99	0.99	497
Weighted avg	0.99	0.99	0.99	497

  - Mô hình phân loại đạt hiệu suất rất cao trên tất cả các chỉ số, với độ chính xác, khả năng nhớ, và F1-score đều gần như hoàn hảo. Điều này cho thấy mô hình rất đáng tin cậy và hiệu quả trong việc phân loại các loại rác khác nhau.
- **Tốc độ thực thi :** 1.6 ~ 1.7s

3.2. Kết quả phần cứng

- Cấu trúc thùng rác thông minh:
  - Nắp tự đóng mở.
  - 3 thanh bập bên với nhiệm vụ phân loại rác.
  - 4 ngăn chứa rác theo thứ tự là giấy, kim loại, nhựa và các loại rác còn lại không thuộc 3 loại rác trên.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Yêu cầu để hoạt động: thùng rác cần phải được cắm điện thông qua adapter 5V-2A.
- Hoạt động của thùng rác thông minh:
    - Bước 1: Nhận diện người đến gần thùng rác để tự động mở/đóng nắp.

      Trên nắp của thùng các có trang bị cảm biến siêu âm đã được cấu hình. Khi người dùng đến gần thùng rác và cách cảm biến siêu âm 50cm. Cảm biến siêu âm sẽ gởi tín hiệu về cho ESP8266. ESP8266 gởi tín hiệu đến mạch điều khiển PWM PCA9685 điều khiển servo nắp mở. Nắp mở 5s và tự động đóng lại

    - Bước 2: Chụp ảnh và nhận diện rác.

      Sau bỏ rác vào trong thùng, rác sẽ nằm ở bập bênh thứ nhất. ESP32 chụp ảnh rác và gửi lên server để xử lý. Kết quả nhận diện sẽ được gởi về ESP8266. Từ đó ESP8266 đưa ra các phương xoay bập bênh để đưa rác vào đúng vị trí phân loại được nhận diện.

    - Bước 3: Phân loại rác.

      ESP8266 gửi tín hiệu cho mạch điều khiển PWM PCA9685 để điều khiển các servo dùng để xoay các bập bênh tương ứng. Đối với từng loại rác được nhận diện, chỉ cần các servo quay theo kịch bản tương ứng thì rác sẽ vào đúng ngăn.

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Kiểm thử khả năng phân loại của thùng rác: thùng tác phân loại đúng với tất cả các kết quả nhận diện được server đưa ra.
- Kiểm thử khả năng chịu được sức nặng của bập bênh: đối với rác có khối lượng nhỏ hơn hoặc bằng 750gr thì bập bênh có thể phân loại tốt, nếu khối lượng rác lớn hơn có thể khiến cho bập bênh bị vỡ hoặc mối keo dính giữa servo và bập bênh bị rớt ra.
- Kiểm thử khả năng lưu trữ: thể tích các ngăn chứa khá nhỏ, vì thế các ngăn sẽ nhanh đầy rác. Có thể cải tiến để tăng thêm khả năng lưu trữ rác của các ngăn.

3.3. Kết quả mobile

- Hoàn thành xong giao diện:
  - Giao diện đăng nhập
  - Giao diện đăng kí
  - Giao diện trang chính
  - Giao diện thông báo
  - Giao diện dữ liệu rác
  - Giao diện điều khiển
  - Giao diện xem thống kê rác
  - Giao diện thông tin tài khoản
  - Giao diện thêm thùng rác
  - Giao diện đổi mật khẩu
- Hoàn thành được các chức năng:
  - Đăng ký, đăng nhập, đăng xuất tài khoản người dùng
  - Thêm thùng rác quản lý
  - Xem thông tin chính về thùng rác quản lý: số lượng rác mỗi loại, dữ liệu rác cụ thể, cảnh báo ngăn rác sắp đầy
  - Xem thông báo
  - Xem dữ liệu rác của mỗi thùng
  - Điều khiển thùng rác đóng mở
  - Xem dự liệu thống kê rác trên toàn hệ thống ở mỗi khu vực
  - Xem thông tin cá nhân, đổi mật khẩu
- APP

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Giao diện đăng ký:
  - Tại đây người dùng (người quản lý môi trường hay các quyền hạn liên quan) sẽ nhập các thông tin bắt buộc như: Tên tài khoản, Số điện thoại liên lạc, địa chỉ email cá nhân, mật khẩu đăng nhập.
  - Sau khi đăng ký thành công người dùng có thể đăng nhập tài khoản vào hệ thống với số điện thoại và mật khẩu đã đăng ký, nếu đăng ký thất bại người dùng cần kiểm tra lại các thông tin cảnh báo đã xuất hiện .
- Giao diện đăng nhập: Đăng nhập vào hệ thống bằng cách nhập số điện thoại và mật khẩu đã đăng ký thành công trước đó.
- Giao diện trang chính:
  - Tại đây đầu tiên người dùng sẽ thấy mục thùng rác đang hiển thị ở góc trên bên trái màn hình, khi bấm vào sẽ hiện ra các danh sách thùng rác đang quản lý, có thể lựa chọn thêm thùng rác nếu cần thiết.
  - Ở góc trên bên phải màn hình là mục User bao gồm các chức năng như xem thông tin tài khoản, đổi mật khẩu, đăng xuất.
  - Ở giữa màn hình phía bên trên là hiển thị ngăn rác sắp đầy cùng với tỉ lệ (%) chổ trống của nó.
  - Ở dưới là hiển thị số lượng rác của mỗi loại: Metal, Plastic, Paper, Cardboard.
  - Ở dưới cùng là thanh bar chuyển hướng đến các chức năng chính khác.
- Giao diện thông báo:
  - Tại đây hiển thị tất cả các thông báo ngăn rác đầy của những thùng rác mà người dùng đang quản lý.
  - Hiển thị thông báo dạng danh sách, một thông báo bao gồm mã thùng rác, ngày giờ, nội dung thông báo
- Giao diện dữ liệu rác:
  - Tại đây hiển thị danh sách mới nhất về các rác thải có trong thùng rác.
  - Một rác thải được hiển thị trong danh sách bao gồm: hình ảnh rác, kết quả nhận diện rác, thời gian vứt rác.
  - Ở góc trên bên trái là mã thùng rác đang hiển thị dữ liệu, có thể chọn thùng rác khác hoặc thêm thùng rác bằng cách nhấn vào nó.
- Giao diện điều khiển:

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

- Tại đây hiển thị hình ảnh rác mới nhất của thùng rác cùng với các nút bấm thực hiện chức năng tắt hay mở thùng rác.
- Ở góc trên bên trái là mã thùng rác đang điều khiển, có thể chọn thùng rác khác hoặc thêm thùng rác bằng cách nhấn vào nó, góc trên bên phải là trạng thái hiện tại của thùng rác.

- Giao diện thống kê:
  - Tại đây hiển thị biểu đồ cột thống kê lượng rác thải trung bình của mỗi loại ở các khu vực trong nội thành Đà Nẵng
  - Dưới đó là danh sách nhận xét lượng rác thải trong thành phố.
- Giao diện thêm thùng rác:
  - Tại đây người dùng sẽ nhập các thông tin cơ bản của thùng rác: mã thùng rác, khu vực, mô tả
  - Thêm thành công khi đã nhập đầy đủ và chính xác thông tin, thất bại khi mã thùng rác không trùng khớp.
- Giao diện xem thông tin các nhân:
  - Tại đây hiển thị thông tin tài khoản :Tên đăng nhập, Số điện thoại, địa chỉ email cá nhân.
  - Và danh sách thông tin các thùng rác của người dùng quản lý.
- Giao diện đổi mật khẩu:
  - Tại đây người dùng có thể đổi mật khẩu của tài khoản sử dụng để đăng nhập bằng cách nhập chính xác các thông tin: mật khẩu hiện tại , mật khẩu mới.
  - Đổi mật khẩu thành công khi nhập đúng mật khẩu hiện tại và ngược lại là thất bại.

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

CHƯƠNG 4. KẾT LUẬN

4.1 Kết luận

4.1.1. Độ chính xác

- Độ chính xác của thuật toán phân loại sử dụng mạng CNN khá cao với ***loss*** và ***accuracy*** trên tập dữ liệu kiểm thử lần lượt sau 23 epochs là 0.0553 và 0.9774

4.1.2. Tốc độ thực thi

Tham số đo	Giá trị	Đơn vị	Ghi chú
Thời gian phản hồi của thùng rác	3 - 5	giây	Tốc độ phụ thuộc vào đường truyền và tình trạng phần cứng
Tốc độ phân loại rác của server	1 - 2	giây	Tốc độ phụ thuộc vào đường truyền mạng khi gửi ảnh từ ESP32-CAM lên server
Sức chứa	32000	cm^3	Có thể mở rộng thêm

4.1.3. Tính ổn định

- Đánh giá độ ổn định theo thang đo từ 1 - 10
- Các bài kiểm tra được thực hiện 20 lần, sau đó thống kê các kết quả để đưa ra mức đánh giá

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

Tham số đo	Đánh giá (1 -10)	Ghi chú
Ổn định phần cứng	6	Để hệ thống khởi động trước vài phút sẽ tăng tính ổn định
Ổn định phần mềm	8	Góc chụp và độ sáng sẽ ảnh hưởng đến tính ổn định
Ổn định kết nối mạng	9	Nên sử dụng mạng kết nối 4G phát từ điện thoại
Ổn định nguồn năng lượng	8	Nên cắm chặt các cổng kết nối để đầu ra nguồn điện ổn đỉnh (5V - 1A) và đảm bảo sạc dự phòng được sạc đầy

4.2 Kết quả đạt được

4.2.1. Về sản phẩm

- Mặt tốt:
  - Hệ thống đáp ứng được các yêu cầu cơ bản đã đặt ra: cảm biến tự động đóng/mở, chụp ảnh, gửi, xử lý phân loại, thông báo rác đầy
  - Chi phí các linh kiện không quá cao
  - Độ chính xác phân loại khá cao trong điều kiện lí tưởng
  - Có khả năng mở rộng trong tương lai
- Mặt hạn chế:
  - Các linh kiện chỉ đạt độ ổn định tương đối, và dễ hỏng trong quá trình sử dụng
  - Kết nối giữa phần mềm và phần cứng chưa đạt ổn định cao nhất
  - Chưa phân loại đa dạng các loại rác

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

4.2.2. Về con người

- Tìm hiểu và ứng dụng những kiến thức đã học trong việc phát triển một dự án kĩ thuật máy tính
- Học được cách kết nối phần cứng và phần mềm
- Phát triển thêm kĩ năng phân chia và làm việc hiệu quả

4.3 Hướng phát triển

- Tiếp tục thu thập dữ liệu để tăng độ chính xác. Mở rộng thêm các lớp phân loại, để phân loại rác chi tiết hơn
- Tìm hiểu nhiều thuật toán có độ chính xác và tốc độ phân loại nhanh hơn
- Nếu có thêm kinh phí thì sẽ nâng cấp các phần cứng tốt hơn để tăng tốc độ xử lý
- Mở rộng đặt thêm nhiều thùng rác và quản lý tất cả các thùng rác trên website

This document is available on studocu
Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)

TÀI LIỆU THAM KHẢO

[1] Thông số Arduino Uno R3, ESP32-CAM, ESP8266 - https://nshopvn.com/

[2] Nguyên tắc hoạt động Arduino Uno R3, ESP32-CAM, ESP8266 http://arduino.vn/

[3] Restful API: Complete Beginner to APIs - https://medium.com/

[4] WebSocket: WebSocket với python - https://codelearn.io/sharing/lap-trinh-web-socket-voi-python

[5] Django: Get Started with Django Channels https://blog.heroku.com/in_deep_with_django_channels_the_future_of_real_time_apps_in_django

[6] Django Rest Framework: Xây dựng API với Django Rest Framework https://viblo.asia/p/xay-dung-api-voi-django-rest-framework-Do754PXJ5M6

[7] Mạng CNN: Tìm hiểu về Convolutional Neural Network https://thanhvie.com/tim-hieu-ve-mang-no-ron-tich-chap-convolutional-neural-networks/

[8] RestNet50: Deep Residual Learning for Image Recognition, *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun*- https://arxiv.org/abs/1512.03385/

[9] Transfer Learning: A comprehensive Survey on Transfer Learning*, Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu*- https://arxiv.org/abs/1911.02685

Downloaded by Minh Jenh (s2KylerJoy77@ndt6.asia)