|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A blue and white logo with arrows  Description automatically generatedA blue and yellow logo  Description automatically generatedTRƯỜNG ĐẠI HỌC BÁCH KHOA**  **KHOA CÔNG NGHỆ THÔNG TIN**      **BÁO CÁO ĐỒ ÁN**  **PBL5 - KỸ THUẬT MÁY TÍNH**      **CHUỒNG GÀ THÔNG MINH DÀNH CHO HỘ GIA ĐÌNH**    **CÁN BỘ DOANH NGHIỆP HƯỚNG DẪN: Trần Phương Nam GIẢNG VIÊN ĐỒNG HƯỚNG DẪN: TS. Ninh Khánh Duy**     |  |  | | --- | --- | | **STT NHÓM: 04**  **HỌ VÀ TÊN SINH VIÊN** | **LỚP HỌC PHẦN ĐỒ ÁN** | | **Lê Ngọc Hạnh** | **21TCLC\_DT1** | | **Nguyễn Đức Ngọc Lam** | **21TCLC\_DT1** | | **Vũ Xuân Hoàng** | **21TCLC\_DT1** | | **Lê Phan Phú Việt** | **21TCLC\_DT1** |     **ĐÀ NẴNG, 06/2024** |

**TÓM TẮT ĐỒ ÁN**

Với việc thành phố ngày càng được mở rộng, việc chăn nuôi gà quy mô nhỏ trong những hộ gia đình sống ở thành phố lớn đã không còn quá xa lạ. Tuy nhiên, do cường độ làm việc của người dân thành phố cao, thời gian dành cho việc chăn nuôi từ đó thu hẹp lại cần đến những giải pháp tự động hóa giải phóng sức người. Do đó, chúng em đề xuất và tiến hành xây dựng đồ án với đề tài “Chuồng gà thông minh dành cho hộ gia đình” với mong muốn giúp việc chăn nuôi gà tốn ít thời gian, công sức nhất có thể. Nhóm đã ứng dụng công nghệ AI trong việc phát hiện, đếm số gà; API trong việc giao tiếp giữa ứng dụng và server, ESP32 – Cam để chụp ảnh, xây dựng ứng dụng di động để hiển thị kết quả quá trình và sử dụng điện toán đám mây để thiết lập server. Sau khi nghiên cứu và thử nghiệm, hệ thống đã hoạt động tốt. Tuy nhiên, còn một số điểm thiếu sót nên hệ thống vẫn chưa hoàn hảo. Nhóm sẽ tiếp tục phát triển và hoàn thiện trong tương lai.

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |
| --- | --- | --- |
| **Sinh viên** | **Nhiệm vụ** | **Hoàn thành** |
| **Lê Ngọc Hạnh** | **Phân công công việc, đảm bảo tiến độ đồ án** | **** |
| **Tìm hiểu, triển khai phần phát hiện gà** | **** |
| **Tìm kiếm và chuẩn bị phần cứng** | **** |
| **Hỗ trợ xây dựng cơ sở dữ liệu** | **** |
| **Hỗ trợ xây dựng server** | **** |
| **Ghép nối và thử nghiệm sản phẩm** | **** |
| **Viết báo cáo** | **** |
|  | **** |
| **Nguyễn Đức Ngọc Lam** | **Tìm hiểu, triển khai phần server** | **** |
| **Xây dựng cơ sở dữ liệu** | **** |
| **Xây dựng server** | **** |
| **Ghép nối và thử nghiệm sản phẩm** | **** |
| **Viết báo cáo** | **** |
| **Vũ Xuân Hoàng** | **Xây dựng ứng dụng mobile app** | **** |
| **Xây dựng cơ sở dữ liệu** | **** |
| **Hỗ trợ xây dựng server** | **** |
| **Ghép nối và thử nghiệm sản phẩm** | **** |
| **Viết báo cáo** | **** |
| **Lê Phan Phú Việt** | **Tìm kiếm và chuẩn bị phần cứng** | **** |
| **Xây dựng phần cứng** | **** |
| **Tìm kiếm và gán nhãn dữ liệu cho AI** | **** |
| **Ghép nối và thử nghiệm sản phẩm** | **** |
| **Viết báo cáo** | **** |
| **Làm slide** | **** |

**MỤC LỤC**

[1. Giới thiệu 7](#_Toc168927361)

[1.1 Thực trạng sản phẩm 7](#_Toc168927362)

[1.2 Các vấn đề cần giải quyết 7](#_Toc168927363)

[1.3 Đề xuất giải pháp tổng quan 7](#_Toc168927364)

[2. Giải pháp 9](#_Toc168927365)

[2.1 Giải pháp về phần cứng và truyền thông 9](#_Toc168927366)

[2.1.1 Sơ đồ tổng quan hệ thống 9](#_Toc168927367)

[2.1.2 Sơ đồ hoạt động tổng quan 10](#_Toc168927368)

[2.1.3 Linh kiện sử dụng 10](#_Toc168927369)

[2.2 Chi tiết module camera và cảm biến nhiệt độ 13](#_Toc168927370)

[2.2.1 Sơ đồ lắp mạch ESP32 – Cam kết hợp cảm biến nhiệt AMG8833 13](#_Toc168927371)

[2.2.2 Cách thức hoạt động của module ESP32 – Cam và cảm biến nhiệt độ AMG8833 16](#_Toc168927372)

[2.3 Chi tiết module tự động cho ăn và đo khối lượng đồ ăn, thức uống 17](#_Toc168927373)

[2.3.1 Thiết kế mô hình tự động cho gà ăn 17](#_Toc168927374)

[2.3.2 Thiết kế mạch cho module tự động cho ăn và đo khối lượng thức ăn, nước uống 19](#_Toc168927375)

[2.3.3 Cách thức hoạt động 20](#_Toc168927376)

[2.4 Truyền thông 21](#_Toc168927377)

[2.4.1 RESTful API 21](#_Toc168927378)

[2.4.2 Giới thiệu về Flask 22](#_Toc168927379)

[2.5 Giải pháp TTNT/KHDL 22](#_Toc168927380)

[2.5.1 Giới thiệu về YOLOv8 22](#_Toc168927381)

[2.5.2 Kiến trúc YOLOv8 23](#_Toc168927382)

[2.5.3 Xương sống (Backbone) 24](#_Toc168927383)

[2.5.4 Cổ (Neck) 27](#_Toc168927384)

[2.5.5 Head 27](#_Toc168927385)

[2.5.6 Giải pháp dự đoán gà bệnh 28](#_Toc168927386)

[2.5.7 Đo lượng thức ăn của gà 30](#_Toc168927387)

[(1) Xác định lượng thức ăn tiêu thụ trung bình 30](#_Toc168927388)

[(2) Xác định gà bị bệnh qua lượng thức thức ăn tiêu thụ trung bình 30](#_Toc168927389)

[2.5.8 Module cảm biến nhiệt độ 31](#_Toc168927390)

[(1) Giới thiệu về AMG8833 31](#_Toc168927391)

[(2) Sử dụng AMG8833 để xác định nhiệt độ của từng con gà 1 cách tương đối 33](#_Toc168927392)

[2.6 Cấu trúc Server 33](#_Toc168927393)

[2.6.1 API 33](#_Toc168927394)

[2.6.2 Database 33](#_Toc168927395)

[2.6.3 Hệ thống xử lý dữ liệu 33](#_Toc168927396)

[2.7 Giải pháp ứng dụng di động 33](#_Toc168927397)

[2.7.1 Phát triển bài toán 33](#_Toc168927398)

[2.7.2 Công nghệ sử dụng 33](#_Toc168927399)

[2.7.3 Biểu đồ usecase hệ thống 34](#_Toc168927400)

[3. Kết quả 34](#_Toc168927401)

[3.1 Phát hiện gà 34](#_Toc168927402)

[3.1.1 Tập dữ liệu 34](#_Toc168927403)

[3.1.2 Huấn luyện 34](#_Toc168927404)

[3.1.3 Kết quả giải pháp phát hiện gà 36](#_Toc168927405)

[3.2 Nâng cao khả năng nhận diện gà với ảnh chụp từ ESP32-CAM 44](#_Toc168927406)

[3.2.1 Tập dữ liệu 44](#_Toc168927407)

[3.2.2 Huấn luyện 45](#_Toc168927408)

[3.2.3 Kết quả 47](#_Toc168927409)

[3.2.4 Đánh giá tổng quan về quán trình huấn luyện mô hình 52](#_Toc168927410)

[3.2.5 Kết Luận 53](#_Toc168927411)

[3.2.6 Hình ảnh nhận diện gà từ ESP32-CAM với model đã train. 53](#_Toc168927412)

[3.3 Giải pháp tự động cho gà ăn và cân khối lượng thức ăn, nước uống 53](#_Toc168927413)

[3.3.1 Triển khai thực tế 54](#_Toc168927414)

[3.3.2 Kết quả 56](#_Toc168927415)

[3.4 Giải pháp nhận diện gà bệnh 58](#_Toc168927416)

[3.4.1 Dữ liệu 58](#_Toc168927417)

[3.4.2 Kết quả giải pháp nhận diện gà bệnh 58](#_Toc168927418)

[3.5 Server 58](#_Toc168927419)

[3.5.1 API 58](#_Toc168927420)

[3.5.2 Tốc độ thực thi hệ thống 60](#_Toc168927421)

[3.6 Ứng dụng di động 62](#_Toc168927422)

[3.6.1 Giải pháp 62](#_Toc168927423)

[3.6.2 Thiết kế 62](#_Toc168927424)

[3.6.3 Kết quả 65](#_Toc168927425)

[4. Kết luận 66](#_Toc168927426)

[4.1 Đánh giá 66](#_Toc168927427)

[4.2 Hướng phát triển 66](#_Toc168927428)

[5. Danh mục tài liệu tham khảo 66](#_Toc168927429)

# Giới thiệu

## Thực trạng sản phẩm

Hiện nay, trên thế giới đã xuất hiện nhiều sản phẩm hỗ trợ giúp tự động hóa quy trình chăn nuôi gà của những trang trại gà truyền thống bao gồm các tính năng như đếm số lượng gà, tự động cho gà ăn hay gửi thông tin gà về cho chủ thông qua ứng dụng web/di động…Tuy nhiên, những sản phẩm này hiện chỉ áp dụng cho những trang trại gà quy mô lớn với giá thành cao tương xứng. Do đó, chúng em tiến hành thử nghiệm với đề tài này nhằm tìm ra giải pháp cho những hộ gia đình với quy mô chăn nuôi nhỏ với chi phí thấp hơn.

## Các vấn đề cần giải quyết

* Cần có thiết bị phần cứng để thu nhận dữ liệu hình ảnh gà, khối lượng thức ăn, nước uống mà gà đã tiêu thụ cũng như tự động cho gà ăn.
* Phát hiện và đếm số lượng gà.
* Phát hiện gà bệnh và đưa ra cảnh báo.
* Cần ứng dụng để người dùng nhận dữ liệu số lượng gà, lượng thức ăn, nước uống gà tiêu thụ cũng như những cảnh báo khi có gà bị bệnh.
* Hệ thống chạy theo thời gian thực.

## Đề xuất giải pháp tổng quan

**Bảng 1: Đề xuất giải pháp tổng quan**

|  |  |
| --- | --- |
| **Vấn đề** | **Giải pháp đề xuất** |
| Phần cứng | ESP32 – Cam, cảm biến nhiệt độ, cảm biến cân nặng, động cơ servo. |
| Phát hiện và đếm số lượng gà | Xây dựng và huấn luyện model phát hiện gà, số lượng gà là số gà phát hiện được.  Thử nghiệm với các model: YoloV7, YoloV8,…  Huấn luyện trên Kaggle. |
| Phát hiện gà bệnh | Kết hợp dữ liệu từ cảm biến nhiệt độ và dữ liệu thức ăn, nước uống gà tiêu thụ để đưa ra cảnh báo gà bệnh. |
| Ứng dụng để theo dõi gà | Xây dựng ứng dụng điện thoại.  Người dùng có thể xem số lượng gà trong chuồng và nhận cảnh báo khi thiếu gà.  Có tính năng cho phép xem dữ liệu thức ăn, nước uống mà gà tiêu thụ.  Cảnh báo khi phát hiện gà bị bệnh. |
| Server | Viết API bằng Flask.  Sử dụng MongoDB để lưu trữ dữ liệu. Tính toán xử lý dữ liệu nhận được. |

# Giải pháp

## Giải pháp về phần cứng và truyền thông

### Sơ đồ tổng quan hệ thống

**Hình 1: Sơ đồ tổng quan hệ thống**

Hệ thống bao gồm: module ESP32 – Cam để chụp ảnh gà trong chuồng cùng cảm biến nhiệt độ nhằm phát hiện gà bệnh; module tự động cho gà ăn điều khiển bằng ESP32 sử dụng động cơ servo và đo khối lượng thức ăn, nước uống bằng cảm biến cân nặng; thiết bị smartphone dùng để gửi yêu cầu và nhận thông tin về gà từ server và Cloud server để thiết lập server. Thông qua mạng WiFi, smartphone, ESP32 – Cam và ESP32 có thể giao tiếp với server bằng API. API này được lập trình dựa trên Flask Framework.

### Sơ đồ hoạt động tổng quan

**Hình 2: Sơ đồ hoạt động tổng quan**

### Linh kiện sử dụng

**Bảng 2: Linh kiện sử dụng**

|  |  |  |
| --- | --- | --- |
| **Tên linh kiện** | **Hình ảnh** | **Thông số, hoạt động** |
| **ESP32 – Cam** |  | **Thông số kỹ thuật:**  Bộ vi xử lý Xtensa LX6 dual-core 32-bit  Nguồn 3.3V Micro B  Bộ nhớ SRAM: 520 KB  Bộ nhớ Flash: 4 MB  Mạng không dây 802.11 b/g/n  Bluetooth 4.2  34 cổng GPIO  **Thông số Camera:**  Cảm biến hình ảnh OV2640  Độ phân giải 1600x1200 hoặc 1280x1024 tùy cài đặt |
| **ESP32** |  | **Thông số kỹ thuật:**  Bộ vi xử lý Xtensa LX6 dual-core 32-bit  Nguồn 3.3V Micro B  Bộ nhớ SRAM: 520 KB  Bộ nhớ Flash: 4 MB  Mạng không dây 802.11 b/g/n  Bluetooth 4.2  34 cổng GPIO |
| **Cảm biến nhiệt độ AMG8833** |  | **Thông số kỹ thuật:**  64 cảm biến nhiệt độ hồng ngoại riêng lẻ  Phạm vi nhiệt độ: 0 - 80oC  Độ chính xác: 2.5oC  Độ rộng góc nhìn: 60o  Phạm vi hoạt động: tối đa 5m để phát hiện người |
| **Cảm biến cân nặng (Loadcell) và mạch chuyển đổi HX711** |  | **Thông số kỹ thuật cảm biến cân nặng:**  Phạm vi đo: tối đa 10kg  Độ chính xác: 2g  Điện áp hoạt động: 2.6 – 5.25V  **Thông số kỹ thuật mạch chuyển đổi HX711:**  Bộ chuyển đổi Analog/Digital 24-bit  Điện áp hoạt động: 2.7 - 5V |
| **Động cơ servo** | Động Cơ Micro Servo Quay Liên Tục 360 Độ | **Thông số kỹ thuật:**  Góc quay: 180o  Điện áp hoạt động: 4.8 – 6V  Tốc độ: 0.12 giây/60o ở 4.8V |
| **Mạch chuyển USB UART TTL FT232RL** | Mạch chuyển USB UART TTL FT232RL | Thông số kỹ thuật:  Cổng USB 2.0  Cổng Serial UART TTL 3.3V hoặc 5V  Tốc độ chuyển dữ liệu: tối đa 3Mbps  Điện áp hoạt động: 5V |

## Chi tiết module camera và cảm biến nhiệt độ

### Sơ đồ lắp mạch ESP32 – Cam kết hợp cảm biến nhiệt AMG8833

**Hình 2: Sơ đồ ESP32 – Cam và AMG8833 ở chế độ bootmode**

Cách mắc mạch chuyển USB UART TTL FT232RL vào ESP32 – Cam chế độ bootmode:

- Nối chân VCC của FT232RL vào chân 5V của ESP32 – Cam.

- Nối chân GND của FT232RL vào chân GND của ESP32 – Cam.

- Nối chân RXD (hoặc RXI) của FT232RL vào chân U0T của ESP32 – Cam.

- Nối chân TXD của FT232RL vào chân U0R của ESP32 – Cam.

Cách mắc cảm biến nhiệt độ AMG8833 vào ESP32 – Cam chế độ bootmode:

- Nối chân SCL của AMG8833 vào chân GPIO 2 của ESP32 – Cam.

- Nối chân SDA của AMG8833 vào chân GPIO 14 của ESP32 – Cam.

- Nối chân GND của AMG8833 vào chân GND của ESP32 – Cam.

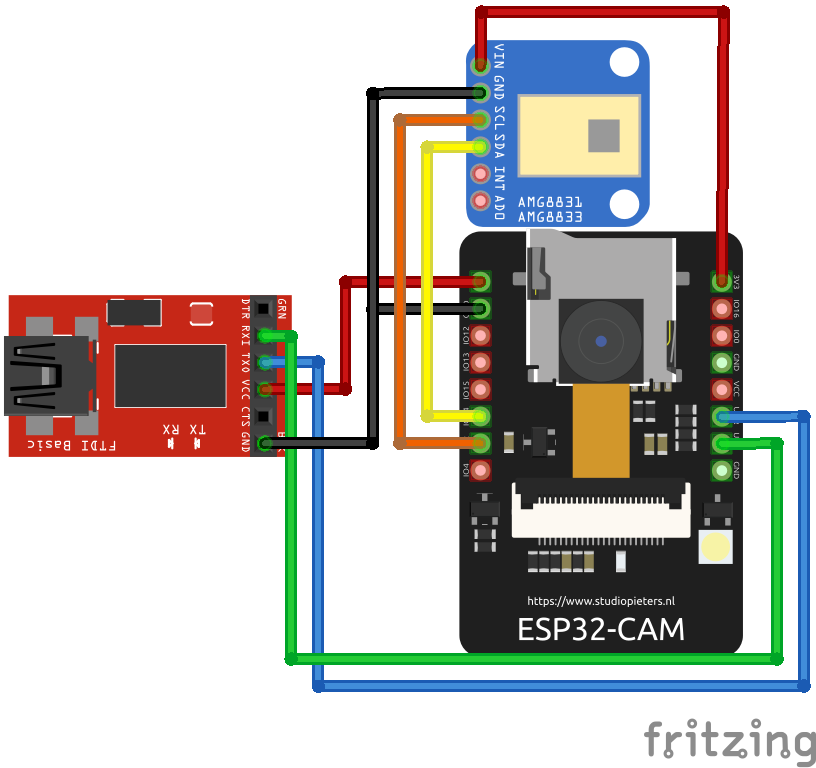
Nối chân IO0 với chân GND trên ESP32 – Cam rồi tiến hành nạp code khi module đang ở trạng thái bootmode.

Sau khi nạp code xong ta sẽ ngắt kết nối USB với máy tính và mắc mạch ở trạng thái hoạt động như sau:

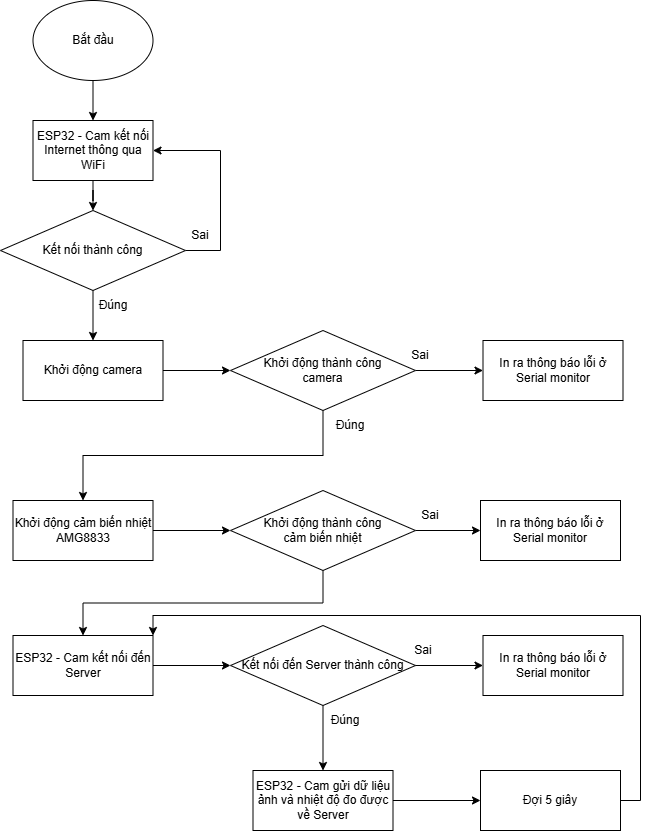
- Ngắt chân IO0 và chân GND của ESP32.

- Nối chân VIN của AMG8833 với chân 3.3V của ESP32.

Kết nối lại module với máy tính dùng để nạp code, module lúc này sẽ hoạt động và gửi ảnh chụp cùng nhiệt độ đo được mỗi 5 giây về máy chủ.

**Hình 3: Sơ đồ lắp mạch ESP32 – Cam và AMG8833 ở chế độ hoạt động**

### Cách thức hoạt động của module ESP32 – Cam và cảm biến nhiệt độ AMG8833

**Hình 4: Sơ đồ khối cách thức hoạt động của module ESP32 – Cam**

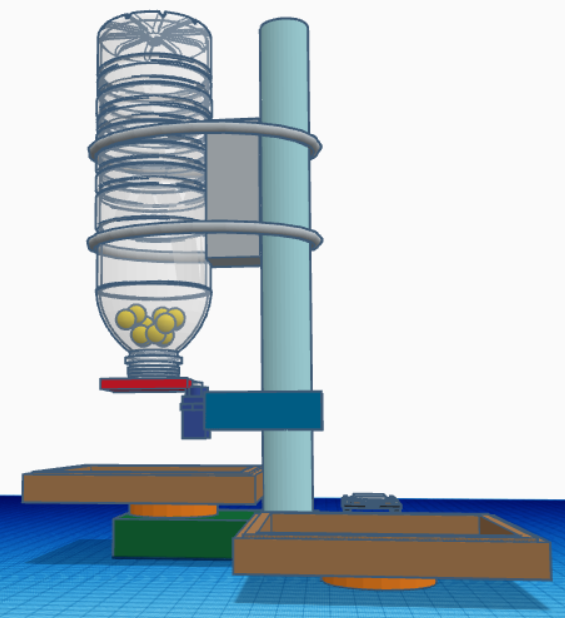
**và cảm biến nhiệt độ**

## Chi tiết module tự động cho ăn và đo khối lượng đồ ăn, thức uống

### Thiết kế mô hình tự động cho gà ăn

**Hình 5: Bản thiết kế 3D của mô hình tự động cho gà ăn**

Mô hình tự động cho gà ăn sử dụng bình nhựa trong 1,5l để đựng thức ăn được cố định bằng dây rút vào ống nhựa, dây rút giúp cho việc điều chỉnh vị trí bình dễ dàng hơn. Ống nhựa được gắn cố định vào khối gỗ chữ nhật được đục một lỗ có đường kính đúng bằng đường kính ống nhựa.

**Hình 6: Bản thiết kế 3D của mô hình tự động cho gà ăn**

Cân sẽ được để dưới máng ăn làm bằng nhựa carton, có hai cân dưới hai máng đựng đồ ăn và thức uống riêng biệt.

### Thiết kế mạch cho module tự động cho ăn và đo khối lượng thức ăn, nước uống

**Hình 7: Sơ đồ lắp mạch cho module tự động cho ăn và cân thức ăn, nước uống**

Nối:

- Chân DOUT của cân thức ăn vào chân GPIO 32.

- Chân SCK của cân thức ăn vào chân GPIO 33.

- Chân DOUT của cân nước uống vào chân GPIO 26.

- Chân SCK của cân nước uống vào chân GPIO 25.

- Dây điều khiển Servo vào chân GPIO 13.

- Dây nguồn của Servo vào chân VIN.  
- Các dây GND vào chân GND tương ứng.

### Cách thức hoạt động

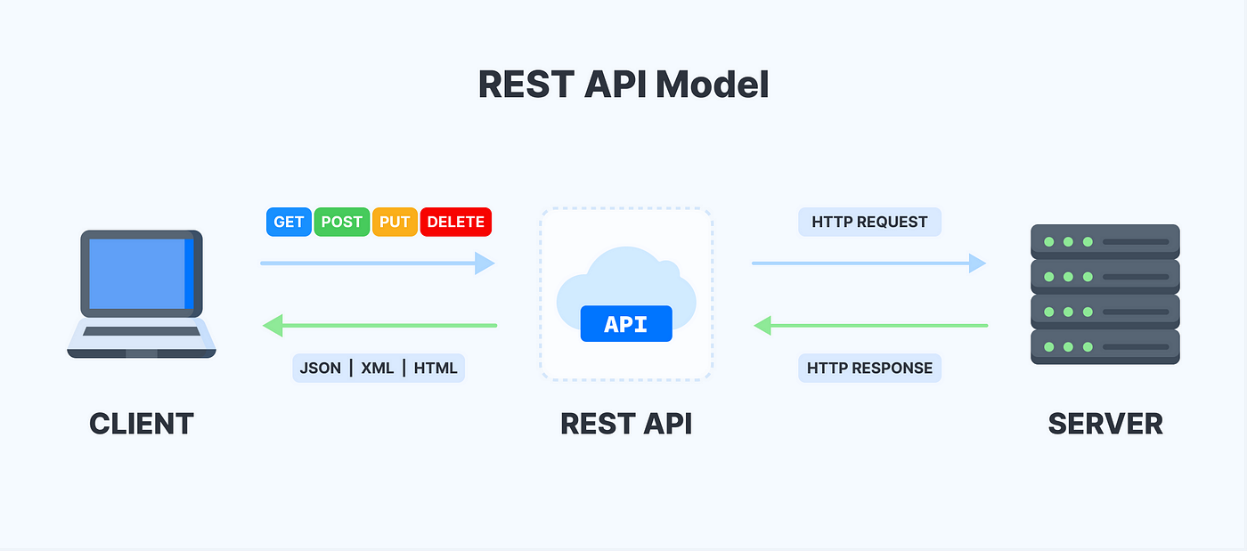
**Hình 8: Sơ đồ khối cách thức hoạt động của module tự động cho ăn**

**và cân khối lượng thức ăn, nước uống**

## Truyền thông

### RESTful API

RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các phần mềm, ứng dụng và dịch vụ web để tạo sự thuận tiện cho việc quản lý các resource. Các tài nguyên hệ thống như tệp văn bản, ảnh, video, âm thanh hay dữ liệu di động là mục tiêu mà nó hướng tới, bao gồm các trạng thái tài nguyên được định dạng và truyền tải qua HTTP. Có thể nói, RESTful API không phải là một loại công nghệ. Nó chỉ là một phương thức tạo ra API và nguyên lý tổ chức nhất định.

* **Các thành phần của RESTful API:**

**Hình 9: Mô hình RESTful API**

API (Application Programming Interface): Là tập hợp các quy tắc và cơ chế mà một ứng dụng hay một thành phần nào đó có khả năng tương tác với một ứng dụng với thành phần khác. API sẽ trả về những kiểu dữ liệu phổ biến như JSON hoặc XML mà ứng dụng của bạn cần sử dụng đến.

REST (Representational State Transfer): Là một dạng chuyển đổi cấu trúc hay kiểu kiến trúc để viết API. Nó có khả năng tạo ra sự tương tác giữa các máy với nhau thông qua phương thức HTTP đơn giản. Chức năng của REST là quy định sử dụng các phương thức HTTP và định dạng URL cho ứng dụng web.

* **Cách thức hoạt động của RESTful API:**

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

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource.

- POST (CREATE): Tạo mới một Resource.

- PUT (UPDATE): Cập nhật thông tin Resource.

- DELETE (DELETE): Xóa một Resource.

### Giới thiệu về Flask

Flask là một microframework web được viết bằng ngôn ngữ lập trình Python. Nó được thiết kế với sự đơn giản, linh hoạt và dễ sử dụng, giúp các lập trình viên dễ dàng xây dựng RESTful API và các ứng dụng web từ đơn giản đến phức tạp.

Ưu điểm của Flask là :

- Đơn giản, dễ sử dụng với cấu trúc đơn giản và mã nguồn dễ đọc, dễ tiếp cận.

- Linh hoạt – không bị giới hạn bởi quy ước hay cấu trúc cụ thể.

- Tính mở rộng cao với nhiều thư viện và tiện ích mở rộng bên thứ ba giúp thêm chức năng cho ứng dụng web.

- Dễ bảo trì: với mã nguồn đơn giản và dễ hiểu, việc bảo trì và sửa lỗi sẽ dễ dàng hơn.

## Giải pháp TTNT/KHDL

### Giới thiệu về YOLOv8

YOLOv8 là một mô hình phát hiện đối tượng mạnh mẽ, linh hoạt và dễ sử dụng với nhiều ưu điểm như tốc độ cao, độ chính xác cao và khả năng tùy chỉnh. Nhờ những ưu điểm này, YOLOv8 được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau.

YOLOv8 sử dụng kiến trúc mạng nơ-ron cuộn (CNN) để trích xuất các đặc điểm từ hình ảnh.

Sau đó, các đặc điểm này được sử dụng để dự đoán vị trí và lớp của các đối tượng trong hình ảnh.

YOLOv8 sử dụng nhiều kỹ thuật tiên tiến như CSPNet, Path Aggregation Network (PANet) và Spatial Attention Module (SAM) để cải thiện hiệu suất của mô hình

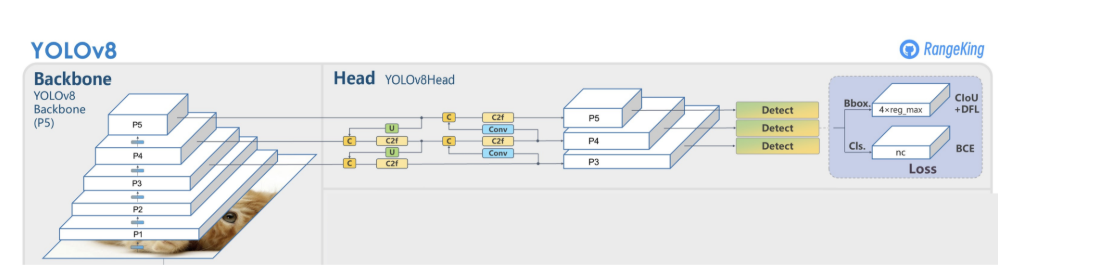
### Kiến trúc YOLOv8

Yolov8 sử dụng kiến trúc của Yolov4 để làm kiến trúc nền tảng và cải tiến

* Backbone: Sử dụng CSPDarknet53 làm backbone, có thể thay thế bằng các backbone CNN khác tương thích.
* Neck: Kết hợp các giai đoạn downsampling và upsampling với các khối PANet và SAM để trích xuất và kết hợp đặc trưng đa cấp.
* Head: Sử dụng đầu YOLOv5 tiêu chuẩn để dự đoán các hộp khung và lớp.

Head là phần cuối cùng của mô hình, chịu trách nhiệm dự đoán các hộp khung và lớp cho các đối tượng được phát hiện.

Nhiệm vụ của từng phần:

* Backbone trích xuất đặc trưng cơ bản từ ảnh.
* Neck kết hợp thông tin đa cấp từ backbone và tăng cường khả năng tập trung vào các vùng có liên quan.
* Head dự đoán các hộp khung và lớp cho các đối tượng được phát hiện.

**Hình 10: Kiến trúc YOLOv8**

### Xương sống (Backbone)

YOLOv8 sử dụng **CSPDarknet53** làm backbone mặc định, tuy nhiên người dùng có thể thay thế bằng các backbone CNN khác như EfficientNet, MobileNet, v.v.

 CSPDarknet53 là phiên bản cải tiến của mạng nơ-ron tích chập Darknet53, sử dụng kiến trúc **CSPNet** (Cross Stage Partial connections) giúp tăng hiệu quả và độ chính xác của mô hình.

 **Cấu trúc CSPNet** bao gồm các khối **CSPLayer** được sắp xếp theo mô hình encoder-decoder. Mỗi CSPLayer bao gồm:

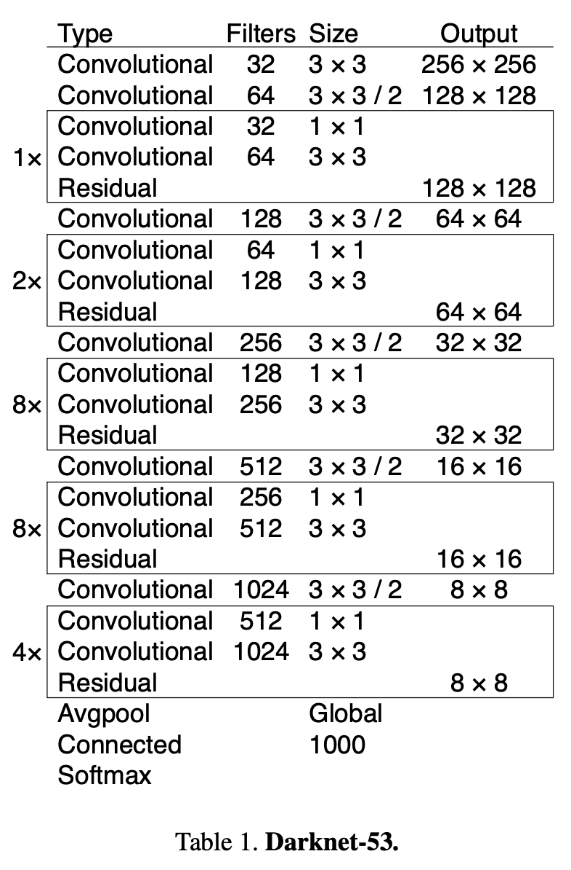
* **Khối Convolutional Neural Network (CNN):** Trích xuất đặc trưng từ ảnh.
* **Khối Cross-Stage Partial connections (CSPC):** Kết hợp thông tin đa cấp từ các giai đoạn downsampling khác nhau, giúp tăng cường khả năng biểu diễn của mô hình.
* **A diagram of a process

  Description automatically generatedHàm kích hoạt Mish:** Thay thế hàm kích hoạt ReLU truyền thống, giúp mô hình học tốt hơn và đạt hiệu suất cao hơn.

**Hình 11: Cấu trúc tổng quan Darknet53**

**Darknet53** là một mạng nơ-ron tích chập (CNN) được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và thị giác máy tính, đặc biệt là trong các ứng dụng phát hiện đối tượng. Mạng nơ-ron này được phát triển bởi Joseph Redmon và Darknet community, và được giới thiệu lần đầu tiên trong bài báo "Darknet: Convolutional Neural Networks for Open Source Object Detection" vào năm 2013.

Cấu trúc:

* Darknet53 có cấu trúc mạng nơ-ron tích chập được sắp xếp theo mô hình encoder-decoder, bao gồm:
  + **Giai đoạn encoder**: Gồm 53 khối Convolutional Neural Network (CNN) được sắp xếp thành 5 giai đoạn downsampling. Mỗi khối CNN bao gồm các hoạt động:
  + **Convolution**: Trích xuất đặc trưng từ ảnh bằng bộ lọc tích chập.
  + **Batch normalization**: Giúp ổn định quá trình huấn luyện và tăng tốc độ hội tụ.
  + **Leaky ReLU**: Hàm kích hoạt phi tuyến tính giúp mô hình học được các hàm phi tuyến tính phức tạp.
  + **Giai đoạn decoder**: Gồm 1 khối upsampling và 3 khối YOLO (You Only Look Once) để dự đoán các hộp khung và lớp cho các đối tượng được phát hiện.

**Hình 12: Cấu trúc chi tiết Darknet53**

**CSPDarknet53** là một mạng nơ-ron tích chập (CNN) được phát triển dựa trên Darknet53, một mô hình CNN nổi tiếng trong lĩnh vực xử lý ảnh và thị giác máy tính. CSPDarknet53 được cải tiến với kiến trúc **Cross Stage Partial connections (CSPNet)** giúp tăng cường hiệu quả và độ chính xác của mô hình trong các nhiệm vụ phát hiện đối tượng.

A diagram of a computer network

Description automatically generated**Hình 13: Cấu trúc CSPDarknet53**

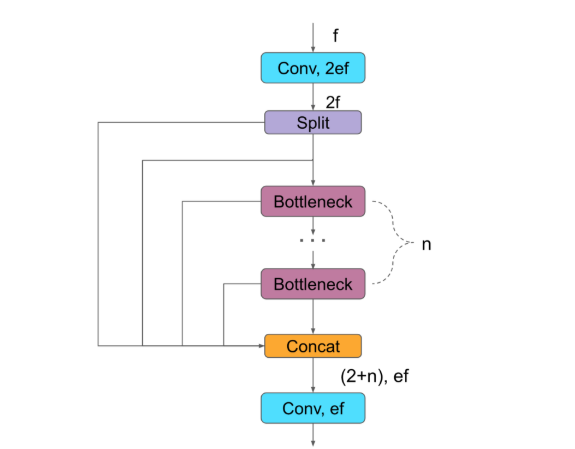
### Cổ (Neck)

**Khác với Yolov5, Yolov8 sử dụng module C2f thay vì kiến trúc truyền thống của YOLO.**

Khối C2f tận dụng tất cả các đầu ra của một đơn vị xử lý trung gian (choke unit) thay vì chỉ sử dụng đầu ra cuối cùng.

Khối C2f bao gồm:

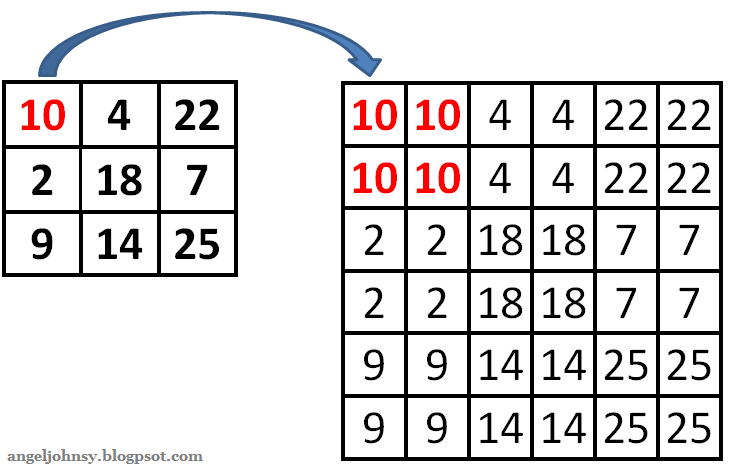
* Hai khối convolutional (giảm số lượng kênh của feature map, thường là giảm một nửa).
* Ba khối convolutional dạng thắt nút (bottleneck).



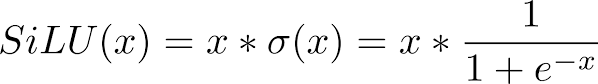
### Head

**Head** (đầu) là thành phần quan trọng thứ ba trong mô hình YOLOv8, nằm sau neck và chịu trách nhiệm **dự đoán các hộp khung và lớp cho các đối tượng** trong ảnh. Head sử dụng các kỹ thuật tiên tiến để đạt được độ chính xác và hiệu suất cao trong nhiệm vụ phát hiện đối tượng.

* **3 lớp CNN:** Mỗi lớp CNN sử dụng các bộ lọc tích chập và hàm kích hoạt SiLU để trích xuất đặc trưng từ các đặc trưng được truyền từ neck.
* **3 lớp upsampling:** Mỗi lớp upsampling sử dụng kỹ thuật **nearest-neighbor interpolation** để tăng kích thước không gian của đặc trưng.
* **Feature fusion module:** Kết hợp thông tin từ các lớp CNN và upsampling tại mỗi giai đoạn.
* **Output layer:** Dự đoán các hộp khung và lớp cho các đối tượng bằng cách sử dụng các kỹ thuật như **anchor boxes** và **classification branch**.

Nearest-neighbor interpolation là một kỹ thuật trong xử lý ảnh được sử dụng để thay đổi kích thước của một hình ảnh.

Hình 14: Nearest-neighbor interpolation

Hàm kích hoạt SiLU (Sigmoid Linear Unit), còn được gọi là Swish, là một hàm kích hoạt trong mạng nơ-ron nhân tạo được giới thiệu bởi các nhà nghiên cứu từ Google. Công thức của hàm SiLU là:

### Giải pháp dự đoán gà bệnh

Dựa vào kĩ thuật nuôi gà, nhóm đã có cho mình một phương án để dự đoán gà bệnh như sau:

**Gà ở trạng thái bình thường:**

Hoạt động:

* Năng động, di chuyển nhiều.
* Ăn uống bình thường, có thể thấy chúng tìm kiếm thức ăn liên tục.

Tướng mạo:

* Mắt sáng, không có dấu hiệu sưng hay chảy nước mắt.
* Lông mượt, óng ánh và không bị rụng nhiều.
* Mào và da quanh mỏ màu đỏ tươi, không bị tím tái hay nhợt nhạt.

Phân:

* Phân có dạng rắn, màu sắc đồng nhất (thường là màu nâu hoặc xanh lá cây).
* Không có mùi hôi thối đặc biệt.

Hành vi:

* Thích chạy nhảy, kiếm ăn, tắm bụi.
* Không có dấu hiệu ho, khò khè hay khó thở.

Nhiệt độ cơ thể của gà bình thường:

* Gà trưởng thành: Nhiệt độ cơ thể bình thường dao động trong khoảng từ 40.6°C đến 41.7°C (105°F đến 107°F).
* Gà con: Nhiệt độ cơ thể có thể cao hơn một chút, khoảng từ 41.1°C đến 42.2°C (106°F đến 108°F).

**Gà ở trạng thái bị bệnh:**

Hoạt động:

* Lờ đờ, ít di chuyển, nằm yên một chỗ.
* Ăn uống kém, bỏ ăn, uống ít nước.

Tướng mạo:

* Mắt lờ đờ, có thể sưng hoặc chảy nước mắt.
* Lông xù, khô, dễ rụng.
* Mào và da quanh mỏ nhợt nhạt, tím tái, hoặc có dấu hiệu viêm nhiễm.

Phân:

* Phân lỏng, màu sắc bất thường (đỏ, trắng, hoặc xanh).
* Có mùi hôi thối khó chịu.

Hành vi:

* Thở khò khè, ho, có thể phát ra tiếng kêu khác thường.
* Đi đứng khó khăn, có thể bị bại liệt một phần.
* Tự tách ra khỏi đàn, ít tương tác với các con khác.

Nhiệt độ cơ thể của gà bị bệnh:

* Khi gà bị bệnh, nhiệt độ cơ thể có thể tăng hoặc giảm tùy thuộc vào loại bệnh và tình trạng sức khỏe của chúng.
* Sốt: Nếu gà bị nhiễm trùng hoặc bị bệnh do vi khuẩn hoặc virus, nhiệt độ cơ thể có thể tăng lên. Nhiệt độ cơ thể gà bị sốt thường vượt quá 41.7°C (107°F) và có thể lên tới 43.3°C (110°F).
* Hạ nhiệt: Trong một số trường hợp bệnh lý nghiêm trọng, đặc biệt là các bệnh gây suy giảm chức năng của hệ miễn dịch hoặc hệ tuần hoàn, nhiệt độ cơ thể của gà có thể giảm xuống dưới mức bình thường, dưới 40.6°C (105°F).

Nhóm đã sử dụng dữ liệu về hành động và nhiệt độ để có thể đưa ra dự đoán về gà bị bệnh có trong chuồng

### Đo lượng thức ăn của gà

Nhóm sử dụng cảm biến cân nặng (Loadcell) để đo lượng thức ăn, nước uống tiêu thụ và gửi về server:

Diagram of a weight scale

Description automatically generated**Hình ??: Đo khối lượng thức ăn, nước uống và gửi về server**

Từ dữ liệu đã thu thập, xác định sự biến động của dữ liệu gửi về, lấy giá trị chênh lệch giá trị gửi về giữa các khoảng thời gian để xác định lượng tổng thức ăn qua một khoảng thời gian nhất định:

### Xác định lượng thức ăn tiêu thụ trung bình

n : là số ngày khoảng thời gian lấy dữ liệu

Xn (gram) : Tổng lượng thức ăn tiêu thụ tính toán được trong khoảng thời gian n ngày từ dữ liệu gửi về

Xtb : Lượng thức ăn tiêu thụ trung bình trong 1 ngày

Xtb =

### Xác định gà bị bệnh qua lượng thức thức ăn tiêu thụ trung bình

Dựa vào lượng thức ăn trung bình sẽ biết lượng thức ăn tất cả gà trong chuồng tiêu thụ:

Với x : lượng thức ăn tiêu thụ trong ngày:

k: số lượng gà trong chuồng:

x ≥ Xtb : thì tất cả trong chuồng đều ăn uống bình thường so với hằng ngày

x ≤ : có gà trong chuồng bỏ ăn

### Module cảm biến nhiệt độ

Nhóm sử dụng camera hồng ngoại AMG8833 để đo nhiệt độ tương đối:

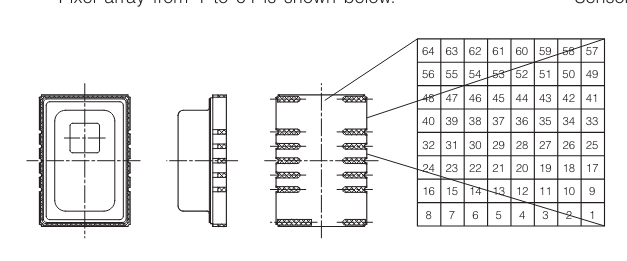
### Giới thiệu về AMG8833

AMG8833 là một cảm biến hình ảnh nhiệt (thermal imaging sensor) được phát triển bởi Panasonic, nổi bật với khả năng phát hiện nhiệt độ và tạo hình ảnh nhiệt với độ chính xác cao. Cảm biến này đã trở thành một công cụ quan trọng trong nhiều ứng dụng công nghiệp, y tế, và các lĩnh vực khác nhờ vào các tính năng ưu việt và khả năng tích hợp dễ dàng.

**Đặc điểm kỹ thuật**

AMG8833 là một cảm biến ma trận nhiệt 8x8, bao gồm tổng cộng 64 pixel nhiệt, mỗi pixel có khả năng đo nhiệt độ riêng biệt. Dữ liệu nhiệt độ từ từng pixel này sau đó được sử dụng để tạo ra một hình ảnh nhiệt của khu vực được quan sát. Một số đặc điểm nổi bật của AMG8833 bao gồm:

* **Độ phân giải**: 8x8 pixel, tạo ra một ma trận hình ảnh nhiệt cơ bản.
* **Phạm vi đo nhiệt độ**: Từ 0°C đến 80°C.
* **Độ chính xác**: ±2.5°C.
* **Tốc độ làm mới**: 10Hz, cho phép cập nhật nhanh chóng các thay đổi về nhiệt độ.



**Đặc tính quang học của AMG8833**

**Trường nhìn (Field of View - FOV)**:

* + **Trường nhìn ngang**: 60°.
  + **Trường nhìn dọc**: 60°.
  + Trường nhìn rộng của AMG8833 cho phép cảm biến quan sát và đo nhiệt độ trên một khu vực rộng, lý tưởng cho các ứng dụng giám sát không gian lớn hoặc phát hiện nhiều đối tượng cùng lúc.

**Độ phân giải không gian (Spatial Resolution)**:

* + AMG8833 có độ phân giải 8x8 pixel, tương ứng với tổng cộng 64 pixel. Mỗi pixel đại diện cho một phần nhỏ trong vùng quan sát và đo nhiệt độ riêng biệt.
  + Độ phân giải này phù hợp cho việc phát hiện sự hiện diện và chuyển động, cũng như giám sát các khu vực lớn mà không cần chi tiết cao.

**Dải phổ nhạy cảm (Spectral Sensitivity Range)**:

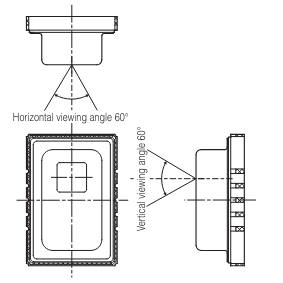
* + Cảm biến nhiệt AMG8833 nhạy cảm với bức xạ hồng ngoại trong dải bước sóng từ 5 µm đến 14 µm.
  + Dải phổ này bao gồm phần lớn bức xạ nhiệt phát ra từ các đối tượng ở nhiệt độ môi trường thông thường, làm cho AMG8833 trở thành công cụ hiệu quả để đo lường nhiệt độ và phát hiện các nguồn nhiệt.

**Độ nhạy nhiệt (Thermal Sensitivity)**:

* + AMG8833 có thể phát hiện các thay đổi nhiệt độ nhỏ, với độ nhạy nhiệt điển hình vào khoảng 0.25°C ở nhiệt độ phòng.
  + Khả năng này giúp cảm biến phát hiện được các biến động nhiệt độ nhỏ, hữu ích trong các ứng dụng yêu cầu độ chính xác cao như y tế và an ninh.

**Phạm vi đo nhiệt độ (Temperature Measurement Range)**:

* + AMG8833 có thể đo nhiệt độ trong khoảng từ 0°C đến 80°C.
  + Dải đo này phù hợp với nhiều ứng dụng giám sát nhiệt độ trong các môi trường khác nhau, từ giám sát nhiệt độ cơ thể con người đến kiểm tra các thiết bị công nghiệp.



### Sử dụng AMG8833 để xác định nhiệt độ của từng con gà 1 cách tương đối

Dựa vào đặc điểm kỹ thuật của AMG8833, khi lấy dữ liệu từ ảnh 8x8 dùng kĩ thuật để tăng kích thước điểm ảnh lên bằng với ảnh chụp từ ESP32-CAM, sau đó đối chiếu với kết quả nhận diện từ model AI nhận diện gà để lấy được vùng điểm ảnh chứa gà phát hiện được, lấy 1 vùng tương ứng trên ảnh hồng ngoại và lấy điểm nhiệt có giá trị lớn nhất.

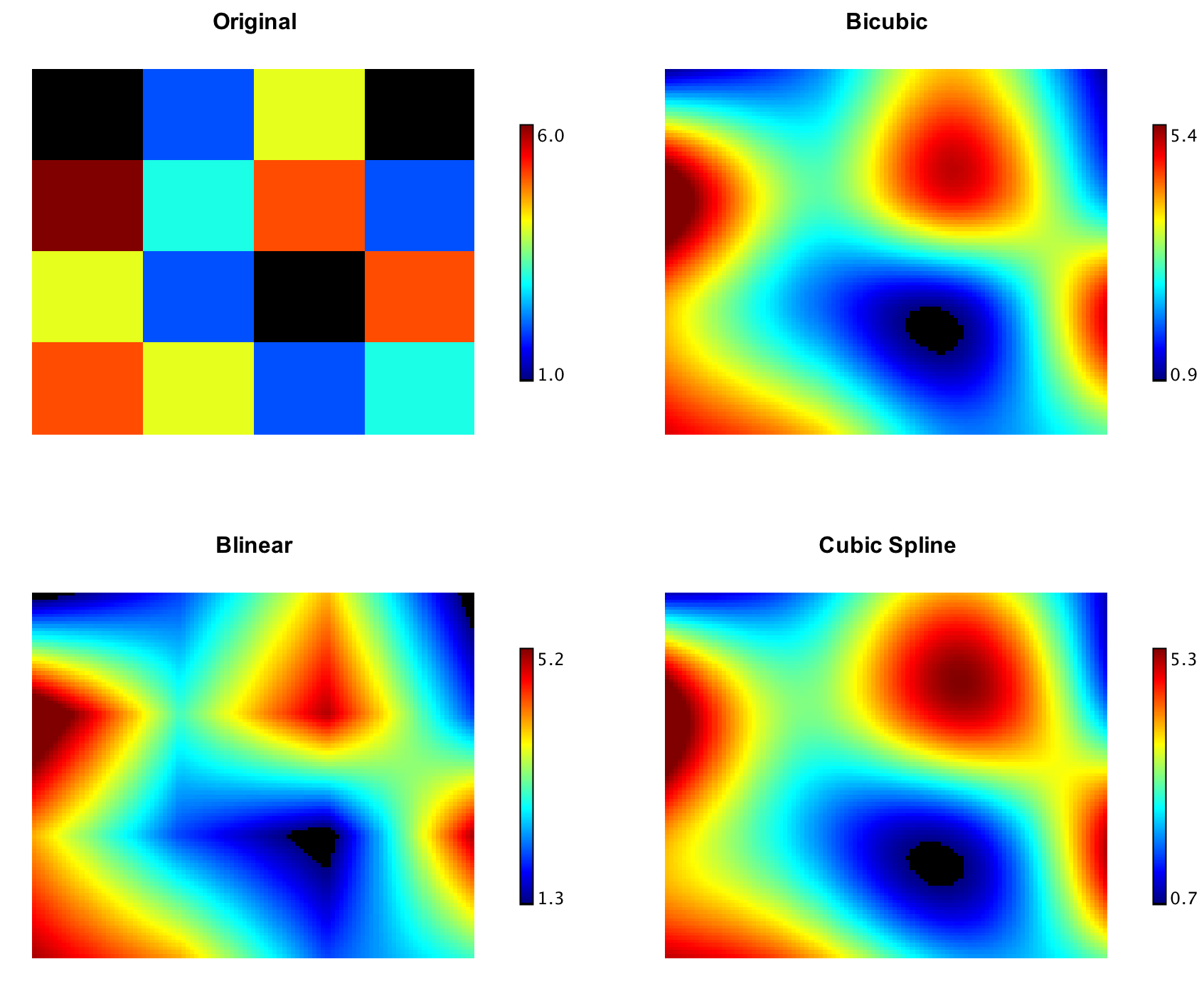
**Bicubic interpolation** là một phương pháp nội suy bậc ba sử dụng hàm đa thức bậc ba để nội suy dữ liệu trong không gian hai chiều (2D). Phương pháp này thường được sử dụng để phóng to hoặc thu nhỏ hình ảnh kỹ thuật số với độ chính xác và chất lượng cao hơn so với các phương pháp nội suy bậc thấp hơn như nội suy gần nhất (nearest neighbor) và nội suy song tuyến tính (bilinear interpolation).

**Nguyên lý của bicubic interpolation**

Bicubic interpolation sử dụng 16 điểm lân cận (4x4) xung quanh điểm cần nội suy. Các điểm này được sử dụng để tính toán giá trị nội suy bằng cách áp dụng các hàm đa thức bậc ba theo cả hai chiều x và y.

Công thức chung cho bicubic interpolation có dạng:





## Giải pháp ứng dụng di động

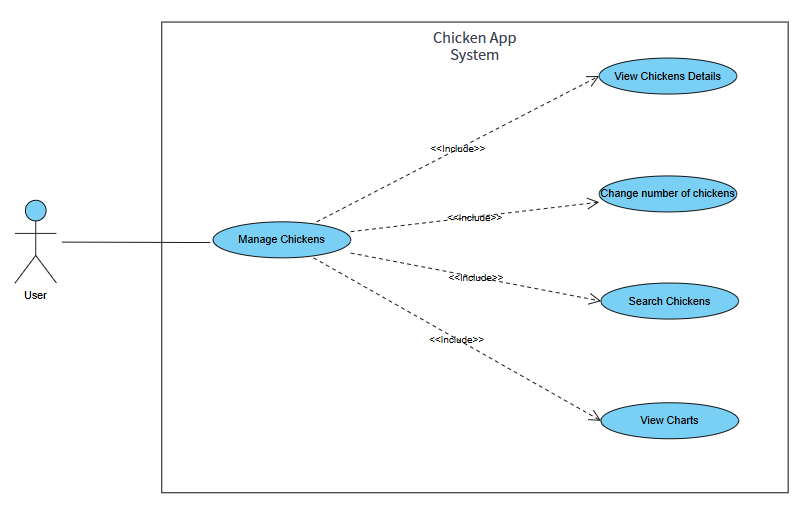
### Phát triển bài toán

Ứng dụng di động quản lý thông tin về số lượng gà, theo dõi và kiểm soát nhiệt độ, lượng thức ăn, nước uống và các thông tin liên quan khác. Ứng dụng cần cung cấp các chức năng cụ thể như chọn ngày để xem bảng thống kê, xem ảnh chụp trong ngày đã chọn, xem chi tiết từng ảnh, xem biểu đồ lượng thức ăn/nước, nhận thông báo về ảnh có nhiệt độ cao, và chỉnh sửa số lượng gà.

### Công nghệ sử dụng

* Sử dụng Android Studio để code ứng dụng Android Native
* Ngôn ngữ lập trình : Java
* Thiết kế giao diện ứng dụng với các nút chức năng ở dưới và màn hình hiển thị ở trên
* Sử dụng SQLite để lưu trữ dữ liệu
* Sử dụng Retrofit để chuyển đổi HTTP API thành Java Interface và xử lý định dạng dữ liệu JSON
* Sử dụng Glide/Picasso để hiển thị ảnh theo URL
* Sử dụng Firebase và Room để lưu trữ dữ liệu

### Biểu đồ usecase hệ thống



# Kết quả

## Phát hiện gà

### Tập dữ liệu

Sử dụng 2500 ảnh thu thập từ Internet,

Phân chia dữ liệu:

* Train: 2000 ảnh
* Test: 250 ảnh
* Val: 250 ảnh

### Huấn luyện

**Các bước tiến hành:**

* Tải thư viện cần thiết:

!pip install ultralytics

* Cấu hình file data.yaml, định nghĩa class và định đường dẫn đến các tập dataset:

train: ../train/images

val: ../val/images

test: ../test/images

nc: 1

names: ['chicken']

* Tiến hành tải pretrain và cấu hình các siêu tham số:

model = YOLO('yolov8x.pt')

model.train(data = '../data.yaml',

           epochs = 100,

            imgsz = height,

            lr0 = 0.01,

            agnostic\_nms = True,

            seed = 42,

            batch = 16,

            patience = 15,

            workers = 8)

Các tham số liên quan:

1. **epochs = 100**:

Số lần toàn bộ tập dữ liệu huấn luyện được truyền qua mô hình. Một epoch là một vòng lặp qua toàn bộ tập dữ liệu. Số epochs càng lớn, mô hình có cơ hội học tốt hơn, nhưng cũng có thể dẫn đến overfitting nếu quá lớn.

1. **imgsz = height**:

Kích thước hình ảnh đầu vào. height có thể là chiều cao của hình ảnh, và hình ảnh sẽ được resize (điều chỉnh kích thước) để phù hợp với mô hình. Điều này ảnh hưởng đến khả năng nhận diện và xử lý hình ảnh của mô hình.

1. **lr0 = 0.01**:

Tốc độ học ban đầu (learning rate). Đây là tốc độ mà tại đó các trọng số của mô hình được cập nhật. Learning rate cao giúp mô hình học nhanh hơn, nhưng nếu quá cao có thể dẫn đến không hội tụ hoặc dao động mạnh. Ngược lại, learning rate quá thấp làm chậm quá trình học. Giá trị 0.01 là một mức trung bình phổ biến, nhưng cần thử nghiệm để tối ưu.

1. **agnostic\_nms = True**:

Non-Maximum Suppression (NMS) không phân biệt lớp (class-agnostic NMS). Khi bật, NMS sẽ loại bỏ các dự đoán trùng lặp bất kể chúng thuộc lớp nào.

1. **seed = 42**:

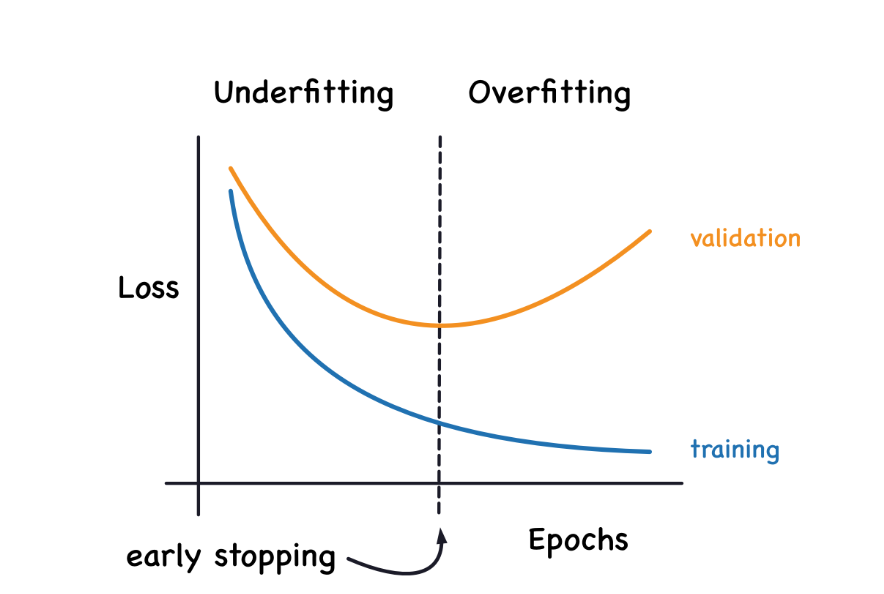
Giá trị ngẫu nhiên được cố định (random seed) để đảm bảo tính tái lập (reproducibility) của kết quả huấn luyện. Giúp các kết quả huấn luyện có thể được lặp lại, thuận tiện cho việc kiểm tra và so sánh các mô hình.

1. **batch = 16**:

Kích thước lô (batch size) là số mẫu được truyền qua mô hình trước khi cập nhật trọng số. Kích thước lô lớn giúp huấn luyện nhanh hơn và ổn định hơn, nhưng đòi hỏi nhiều bộ nhớ hơn. Batch size 16 là một kích thước phổ biến, đặc biệt khi huấn luyện trên GPU.

1. **patience = 15**:

Số epochs chờ đợi trước khi dừng sớm (early stopping) nếu không thấy sự cải thiện trong hàm mất mát. Giúp tránh việc huấn luyện quá lâu mà không thu được lợi ích thêm, tiết kiệm thời gian và tài nguyên.



Hình 15: Kỹ thuật Early Stopping

### Kết quả giải pháp phát hiện gà

Định nghĩa các thông số được dùng để biểu diễn quá trình traing qua đồ thị

 **Loss**: Đo lường độ lệch giữa dự đoán của mô hình và giá trị thực tế. Giá trị loss thấp hơn biểu thị mô hình hoạt động tốt hơn.

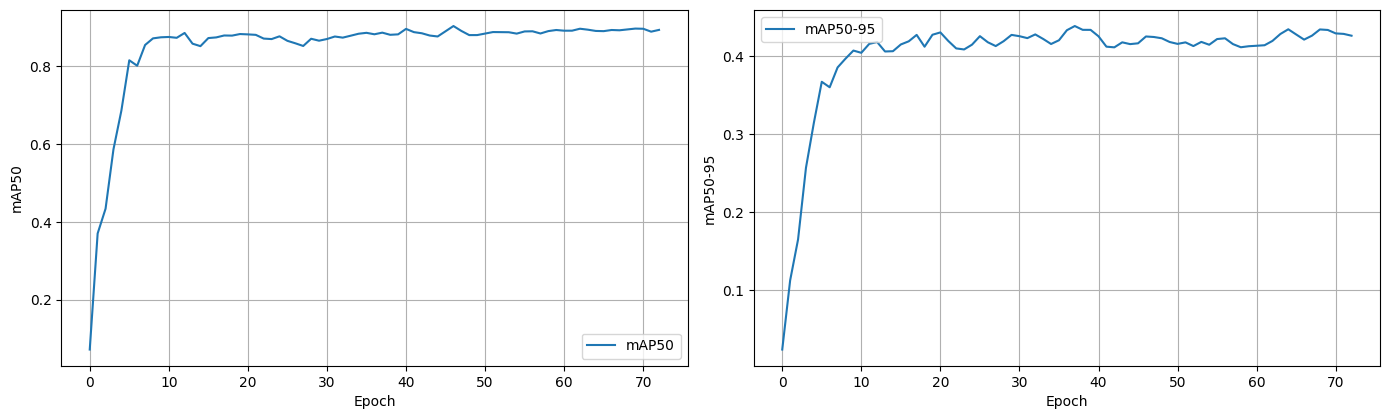
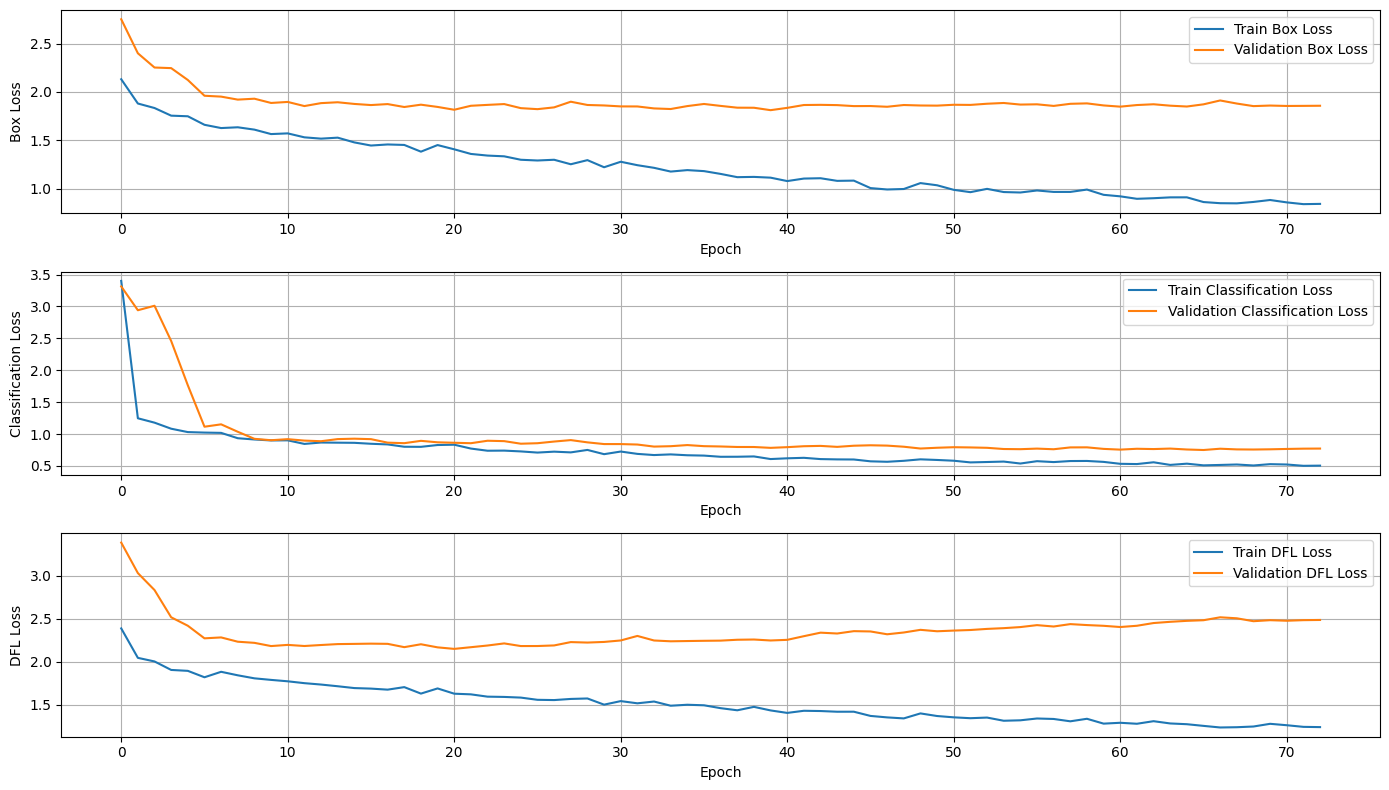
 **DFL Loss**: Là một dạng loss đặc biệt, có thể là một phần của tổng loss. DFL loss thấp hơn thường biểu thị hiệu suất tốt hơn.

 **MPA (Mean Pixel Accuracy)**: Được sử dụng trong phân loại ảnh hoặc nhận dạng đối tượng. MPA cao hơn biểu thị độ chính xác cao hơn trong phân loại từng pixel.

 **Precision**: Tỷ lệ giữa số lượng dự đoán đúng là dương tính trên tổng số lượng dự đoán dương tính. Precision cao cho biết ít dương tính giả.

 **Recall**: Tỷ lệ giữa số lượng dự đoán đúng là dương tính trên tổng số lượng dương tính thực tế. Recall cao cho biết ít âm tính giả.

 **F1-Score**: Trung bình điều hòa của precision và recall. F1-Score cao biểu thị sự cân bằng tốt giữa precision và recall.

 **Confidence**: Độ tin cậy của các dự đoán. Độ tin cậy cao biểu thị mô hình tự tin hơn trong các dự đoán của mình.

Hình 16: Đồ thị theo dõi quá trình training

***Nhận xét:***

**Xu hướng Loss và DFL Loss**: Các giá trị loss và dfl\_loss trên có sự ổn định sau vài epoch đầu và có xu hướng giảm dần theo các epoch, tuy sau đó có sự trững lại và tăng nhẹ và đã kết thúc quá trình huấn luyện bằng kĩ thuật early-stop để tránh overfitting ,điều đó biểu thị mô hình đang học tốt từ dữ liệu huấn luyện.

**MPA** cao và ổn định biểu thị mô hình phân loại tốt trên từng pixel.

**(1) Kết quả định tính**

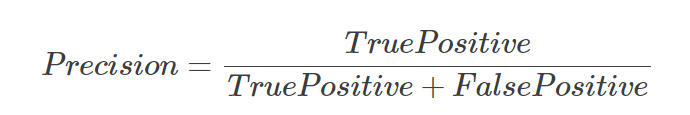
Model có hiệu suất nhận diện cao khi nhận diện ảnh có độ phân giải cao, mang nhiều thông tin của đối tượng

Trong phạm vi chuồng gà vẫn có sự nhầm lẫn và không thể nhận diện khi gà bị che lấp nhau trong một số trường hợp,… chỉ phát hiện được phần ít gà khi sử dụng ảnh từ ESP32CAM

**(2) Kết quả định lượng**

* Sử dụng **AP**( mean Average Precison) để đánh giá độ tin cậy của model. AP là thước đo được sử dụng phổ biến trong các bài toán Object Detection. Ngoài ra còn có các khái niệm như **Precision**, **Recall**, **F1-score**
* **Precision**

Precision cho ta biết **trong số các vật thể mà mô hình dự đoán**, có bao nhiêu phần trăm là **đúng**.

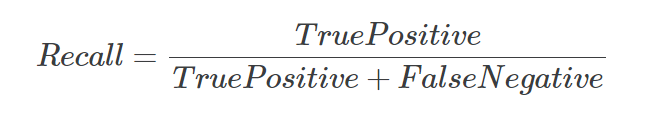


* **True Positive:** Số lượng khung bao dự đoán **chính xác**.
* **False Positive:** Số lượng khung bao dự đoán **sai**.

**Ý nghĩa của Precision:**

* **Precision cao:** Cho biết mô hình có khả năng dự đoán chính xác cao các vật thể, ít dự đoán sai.
* **Precision thấp:** Cho biết mô hình có thể dự đoán sai nhiều vật thể, ảnh hưởng đến độ tin cậy của kết quả.
* **Recall**

**Recall** cho ta biết ***mô hình đã bao quát được bao nhiêu phần trăm các vật thể có trong ảnh.***

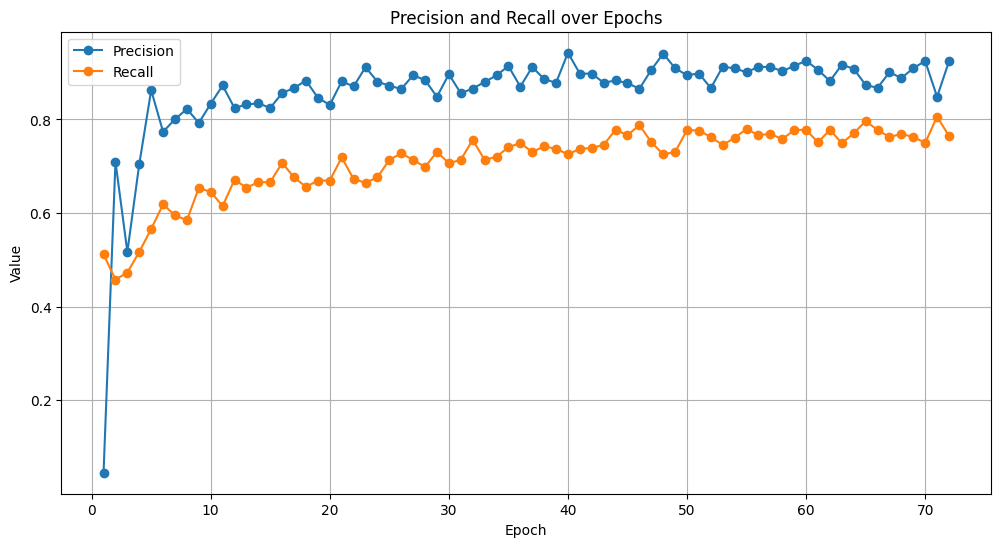


**Trong đó:**

FN (False Negative): Số lượng vật thể thực tế mà mô hình không dự đoán được.

* **Recall cao**: Cho biết mô hình có khả năng thu hồi cao, ít bỏ sót vật thể.
* **Recall thấp**: Cho biết mô hình bỏ sót nhiều vật thể, ảnh hưởng đến độ toàn diện của kết quả.

Đồ thị biểu diễn Precision, Recall theo Epoch trong quá trình huấn luyện

Hình 17: Đồ thị biểu diễn Precision, Recall theo Epoch trong quá trình huấn luyện

Đánh giá:

 Ban đầu, Precision và Recall có thể dao động do mô hình đang học. Tuy nhiên, sau một số epochs, chúng ổn định hơn và cải thiện dần. Điều này là một dấu hiệu tốt cho thấy mô hình đang học hỏi và tối ưu hóa tốt hơn.

 Precision cao hơn Recall trong nhiều epochs cho thấy mô hình thận trọng hơn trong việc đưa ra các dự đoán dương tính.

* **Precison Recall curve**

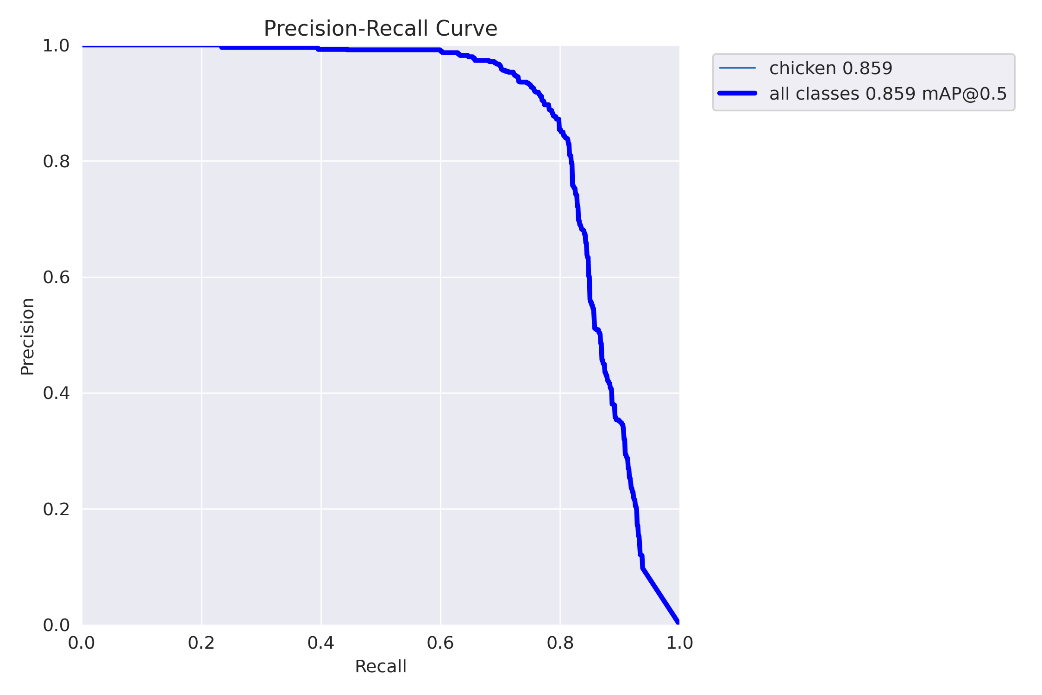
Precision-Recall Curve là một đồ thị được sử dụng để đánh giá hiệu suất của các mô hình phân loại, đặc biệt là trong các bài toán có sự mất cân bằng giữa các lớp.

Đường cong **Precision-Recall Curve** thể hiện mối quan hệ giữa ***precision*** và ***recall*** của một mô hình phân loại khi ngưỡng quyết định thay đổi. Đường cong này được tạo ra bằng cách thay đổi ngưỡng quyết định (threshold) để phân loại các dự đoán, từng bước đếm số lượng true positives, false positives và false negatives, và tính precision và recall tương ứng.

Giả sử có N ngưỡng để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là Pn,Rn, n=1,2,…,N. Precision-Recall curve được vẽ bằng cách vẽ từng điểm có toạ độ (Rn,Pn) trên trục toạ độ và nối chúng với nhau

**Đường cong phản ánh:**

* Một đường cong PR tốt sẽ cong về góc trên bên phải của biểu đồ, đồng nghĩa với việc có precision cao và recall cao ở các ngưỡng quyết định khác nhau.
* Nếu đường cong PR gần với đường chéo từ góc trái dưới đến góc trên trái của biểu đồ, điều này cho thấy mô hình có hiệu suất kém, vì nó không phân biệt giữa true positives và false positives.

Đồ thị biểu diễn Precision-Recall Curve:

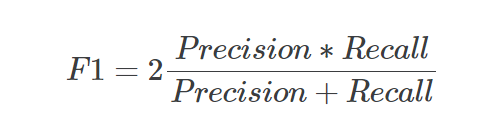
Hình 18: Đường cong Precison-Recall

**Nhận xét:**

Đường cong hướng về góc phải trên của biểu đồ, cho thấy cả Precison và Recall cao điều này cho thấy mô hình có hiệu suất tốt khi phân loại và phát hiện đối tượng

* **F1-score**

**F1-score** (điểm F1) là một thước đo thống kê được sử dụng để đánh giá hiệu suất của mô hình phân loại, đặc biệt là trong lĩnh vực học máy. Nó là **trung bình điều hòa** giữa **Precision** (độ chính xác) và **Recall** (khả năng thu hồi) của mô hình.

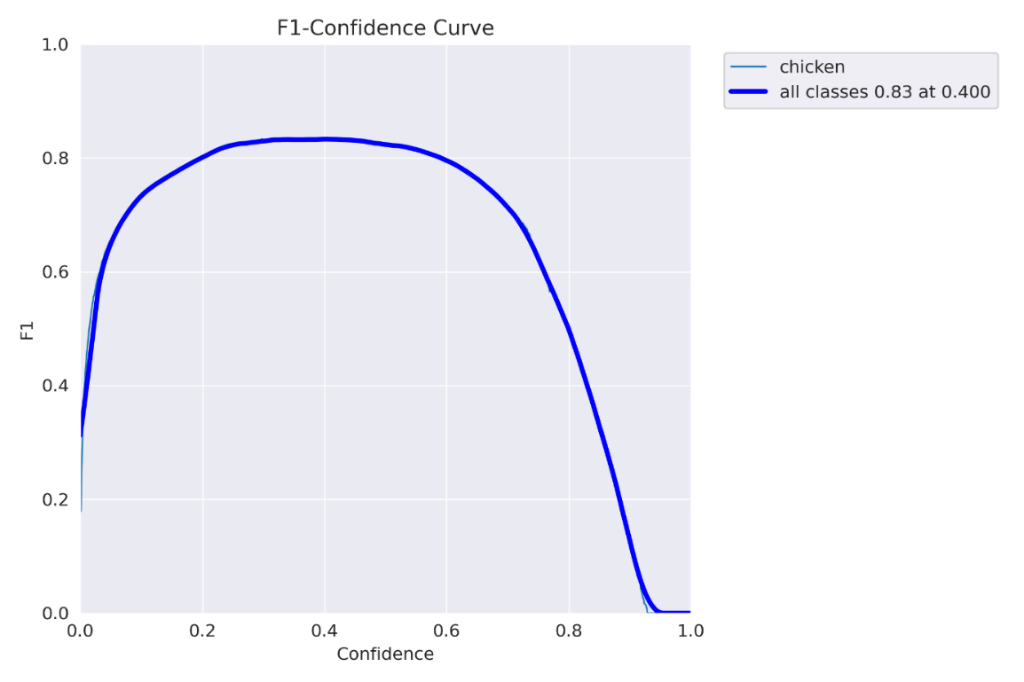
****

**F1-score cao:** Cho biết mô hình có hiệu quả tốt, đạt được Precision và Recall cao.

**F1-score nằm trong khoảng 0 và 1:**

* F1-score = 1: Precision và Recall đều bằng 1, mô hình hoàn hảo.
* F1-score = 0: Precision hoặc Recall bằng 0, mô hình tệ.

Confidence: là một giá trị xác suất mà tại đó mô hình quyết định phân loại một mẫu vào một lớp cụ thể. Trong bối cảnh phân loại nhị phân (chỉ có hai lớp: dương tính và âm tính), mỗi dự đoán của mô hình đi kèm với một xác suất cho biết mức độ tin cậy của mô hình về việc mẫu thuộc vào lớp dương tính.

Đồ thị biểu diễn F1-score trong quá trình huấn luyện:

Hình 19: Đồ thị biểu diễn F1-score trong quá trình huấn luyện

**Nhận xét:**

**Điểm F1 là 0.833**:

* Điểm F1 là 0.833 cho thấy rằng mô hình đang hoạt động rất tốt khi cân bằng giữa Precision và Recall.
* Đây là một hiệu suất tốt, đặc biệt trong các ứng dụng thực tế, một điểm F1 trên 0.8 thường được coi là rất tốt.

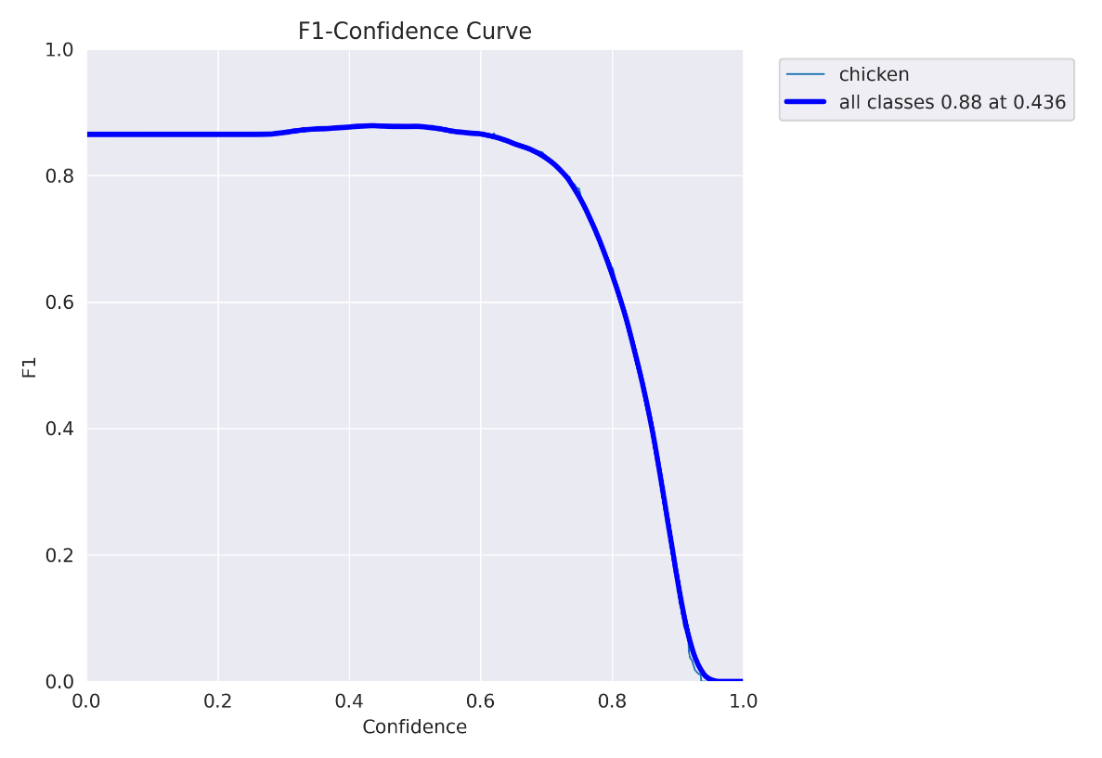
**Ngưỡng Tin Cậy 0.4**:

* Ngưỡng tin cậy 0.4 có thể chỉ ra rằng mô hình ưu tiên việc phát hiện nhiều mẫu dương tính hơn (Recall cao hơn) và chấp nhận một số lượng lớn các dự đoán sai dương tính (Precision có thể bị giảm).

Việc đạt được điểm F1 cao nhất là 0.833 tại ngưỡng tin cậy 0.4 cho thấy mô hình đang hoạt động rất tốt, đặc biệt trong việc đảm bảo rằng các mẫu dương tính được phát hiện đúng

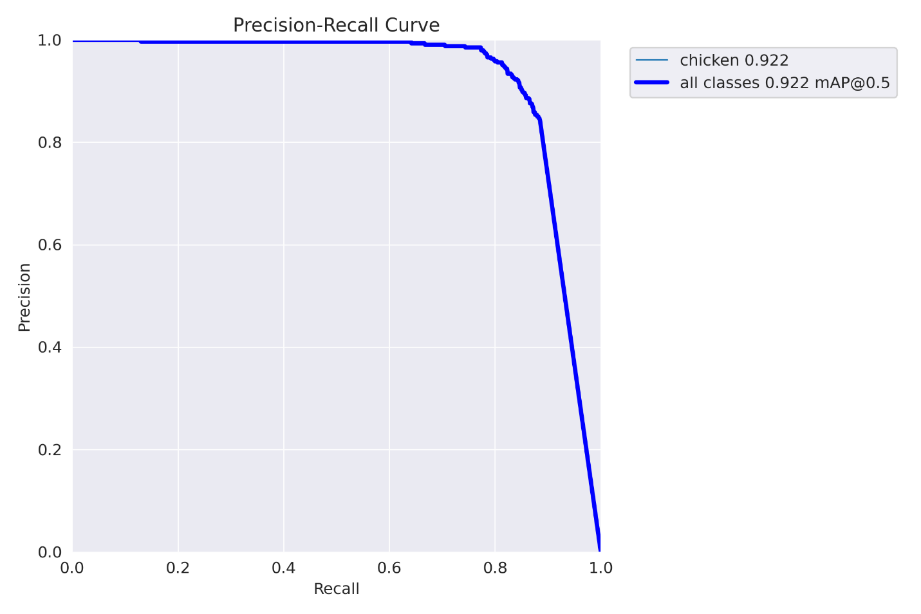
Precision và recall cao cùng với F1-Score cao cho thấy mô hình không chỉ dự đoán đúng nhiều mà còn cân bằng tốt giữa precision và recall.

**(1) So sánh, đánh giá hiệu suất của model tập dữ liệu kiểm tra**

F1-score trên tập test:

Hình 20: F1-score trên tập test

PR-Curve trên tập Test:



Hình 21: Đồ thị PR-Curve trên tập Test

**Nhận xét:**

Trên Tập Kiểm Tra (Test)

* **PR-curve (Precision-Recall curve)**: 0.922
  + Điểm PR-curve là 0.922 là rất cao, cho thấy mô hình duy trì một sự cân bằng rất tốt giữa Precision và Recall trên tập kiểm tra. Điều này cho thấy mô hình có khả năng tổng quát hóa tốt và không chỉ hoạt động tốt trên tập huấn luyện mà còn trên dữ liệu chưa từng thấy.
* **F1-score**: 0.88
  + Điểm F1 là 0.88 cũng là rất tốt, cho thấy mô hình không chỉ có khả năng duy trì sự cân bằng giữa Precision và Recall mà còn cải thiện hiệu suất so với tập huấn luyện.

## Nâng cao khả năng nhận diện gà với ảnh chụp từ ESP32-CAM

### Tập dữ liệu

Dùng ảnh thu được từ ESP32 CAM chụp và ảnh resize lại từ tập dataset đã thu thập trước đó để tang độ nhận diện hơn thì ảnh chụp từ ESP32 CAM có khoảng cách xa, không đủ để nhận diện và phát hiện gà. Tỉ lệ giữa loại ảnh này là 1:1

Tỉ lệ phân chia ảnh: (8:1:1)

* Tập train: 2000 ảnh
* Tập test: 250 ảnh
* Tập Val: 250 ảnh

### Huấn luyện

Thay vì tiếp tục sử dụng pretrain từ Yolov8 như trước đó, lần này nhóm đã sử dụng lại model đã train để nhận biết gà làm pretrain cho quá trình huấn luyện:

Các bước tiến hành:

* Tải thư viện cần thiết:

!pip install ultralytics

* Cấu hình file data.yaml, định nghĩa class và định đường dẫn đến các tập dataset:

train: ../train/images

val: ../val/images

test: ../test/images

nc: 1

names: ['chicken']

* Sử dụng model đã train trước đó để làm pretrain và cấu hình các siêu tham số:

model = YOLO('model\_10.pt')

model.train(data = 'data.yaml',

           epochs=100,

imgsz=height,

optimizer="AdamW",

lr0=0.001,

weight\_decay=0.01,

agnostic\_nms=True,

seed=42,

patience=35,

batch=16,

workers=8)

Các tham số liên quan:

1. **epochs=100**:

Số lần toàn bộ tập dữ liệu huấn luyện được truyền qua mô hình. Một epoch là một vòng lặp qua toàn bộ tập dữ liệu. Số epochs càng lớn, mô hình có cơ hội học tốt hơn, nhưng cũng có thể dẫn đến overfitting nếu quá lớn.

1. **imgsz=height**:

Kích thước hình ảnh đầu vào. height có thể là chiều cao của hình ảnh, và hình ảnh sẽ được resize (điều chỉnh kích thước) để phù hợp với mô hình. Điều này ảnh hưởng đến khả năng nhận diện và xử lý hình ảnh của mô hình.

1. **optimizer="AdamW"**:

Optimizer là thuật toán dùng để cập nhật trọng số của mô hình nhằm giảm thiểu hàm mất mát. AdamW là một biến thể của Adam với việc điều chỉnh thêm trọng số (weight decay) để cải thiện khả năng tổng quát hóa của mô hình.

1. **lr0=0.001**:

Tốc độ học ban đầu (learning rate). Đây là tốc độ mà tại đó các trọng số của mô hình được cập nhật. Giá trị này ảnh hưởng lớn đến quá trình huấn luyện: quá cao có thể khiến mô hình không hội tụ, quá thấp có thể khiến việc huấn luyện quá chậm.

1. **weight\_decay=0.01**:

Trọng số điều chỉnh (weight decay) là một kỹ thuật để tránh overfitting bằng cách thêm một thuật ngữ phạt (penalty term) vào hàm mất mát, nhằm điều chỉnh trọng số của mô hình về giá trị thấp hơn.

1. **agnostic\_nms=True**:

Non-Maximum Suppression (NMS) không phân biệt lớp (class-agnostic NMS). Khi bật, NMS sẽ loại bỏ các dự đoán trùng lặp bất kể chúng thuộc lớp nào.

1. **seed=42**:

Giá trị ngẫu nhiên được cố định (random seed) để đảm bảo tính tái lập (reproducibility) của kết quả huấn luyện.

1. **patience=35**:

Số epochs chờ đợi trước khi dừng sớm (early stopping) nếu không thấy sự cải thiện trong hàm mất mát. Điều này giúp tránh việc huấn luyện quá lâu mà không thu được lợi ích thêm.

1. **batch=16**:

Kích thước lô (batch size) là số mẫu được truyền qua mô hình trước khi cập nhật trọng số. Kích thước lô lớn giúp huấn luyện nhanh hơn, nhưng đòi hỏi nhiều bộ nhớ hơn.

1. **workers=8**:

Số lượng luồng (threads) dùng để tải dữ liệu. Số workers lớn có thể tăng tốc độ tải dữ liệu, đặc biệt hữu ích khi sử dụng nhiều luồng để tải dữ liệu từ đĩa vào bộ nhớ.

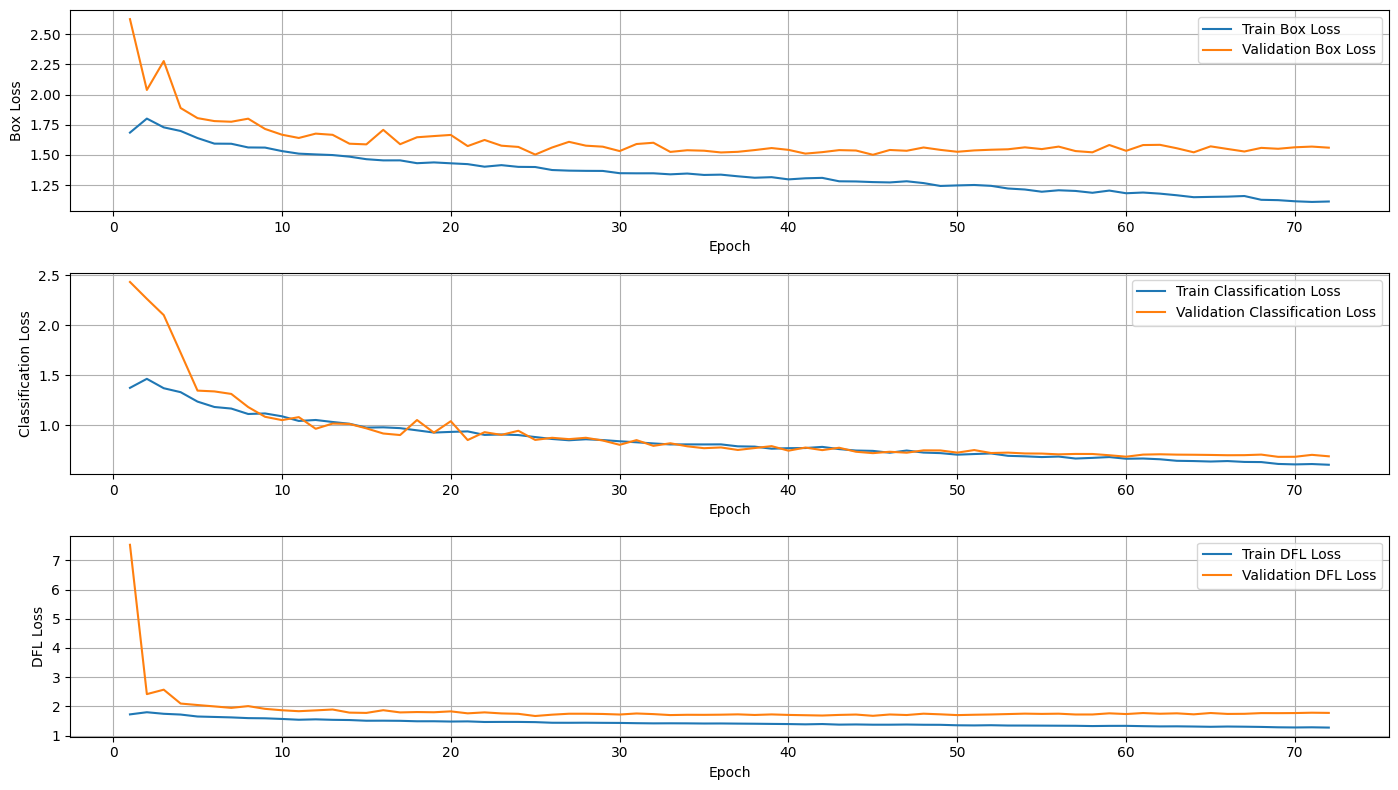
**AdamW** là một thuật toán tối ưu hóa sử dụng adaptive learning rates giống như Adam, nhưng có thêm điều chỉnh trọng số (weight decay) trực tiếp trong quá trình cập nhật. Điều này giúp ngăn chặn overfitting bằng cách làm giảm các trọng số lớn dần trong quá trình huấn luyện.

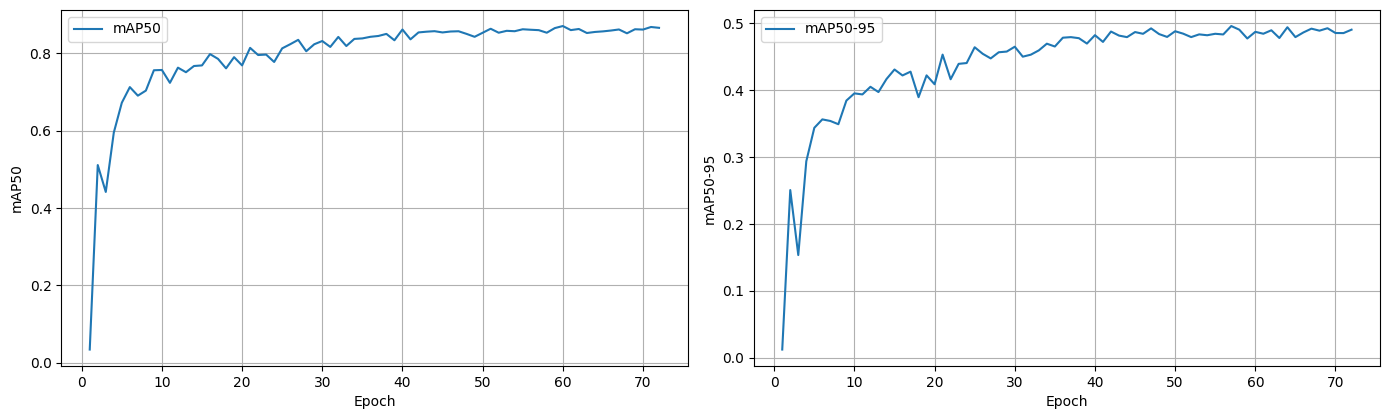
* **lr0 (learning rate)** ảnh hưởng trực tiếp đến tốc độ hội tụ của AdamW.
* **weight\_decay** giúp điều chỉnh trọng số trong quá trình tối ưu hóa, cải thiện khả năng tổng quát hóa của mô hình.

Các siêu tham số khác như **batch size**, **epochs**, và **patience** cũng ảnh hưởng đến hiệu quả của optimizer.

### Kết quả

Đồ thị biểu diễn sự biến thiên của các hàm loss, mPA trong quá trình huấn luyện:





***Nhận Xét:***

**Xu hướng Loss và DFL Loss**: Các giá trị loss và dfl\_loss trên tập Validation có sự biến động ở những epoch đầu nhưng sau đó đã có sự ổn định và hội tụ, sau đó có tăng nhẹ nên đã được cho dừng sớm bằng kĩ thuật early-stop để tránh overfitting, điều đó biểu thị mô hình đang học tốt từ dữ liệu huấn luyện.

**MPA** cao và ổn định biểu thị mô hình phân loại tốt trên từng pixel.

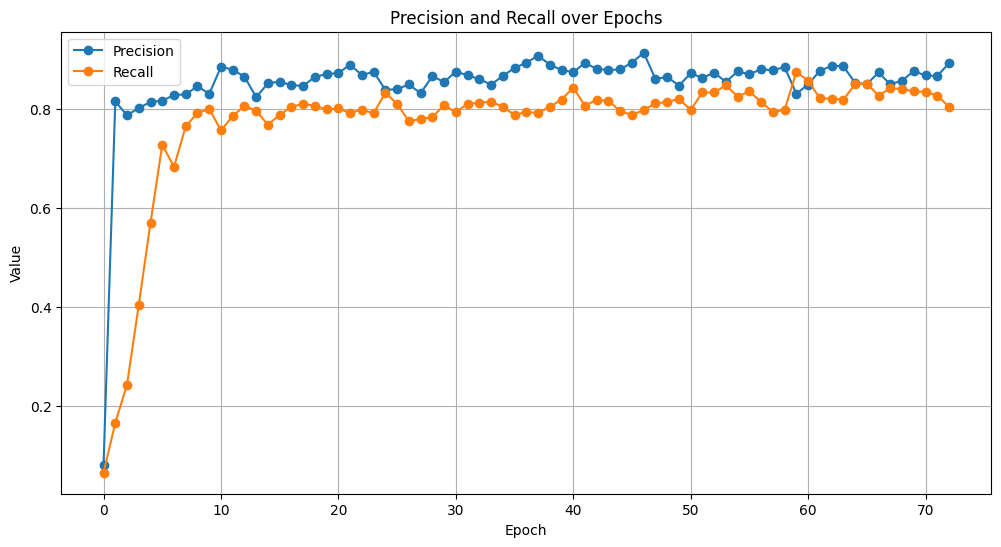
**(1) Kết quả định tính**

Model đã nhận diện được gà thông qua ảnh từ thu được từ ESP32-CAM

Trong phạm vi chuồng gà vẫn có sự nhầm lẫn và không thể nhận diện khi gà bị che lấp nhau trong một số trường hợp,… nhưng tổng thể đã có thể nhận được và cho ra kết quả về số lượng gà trong chuồng đúng

**(2) Kết quả định lượng**

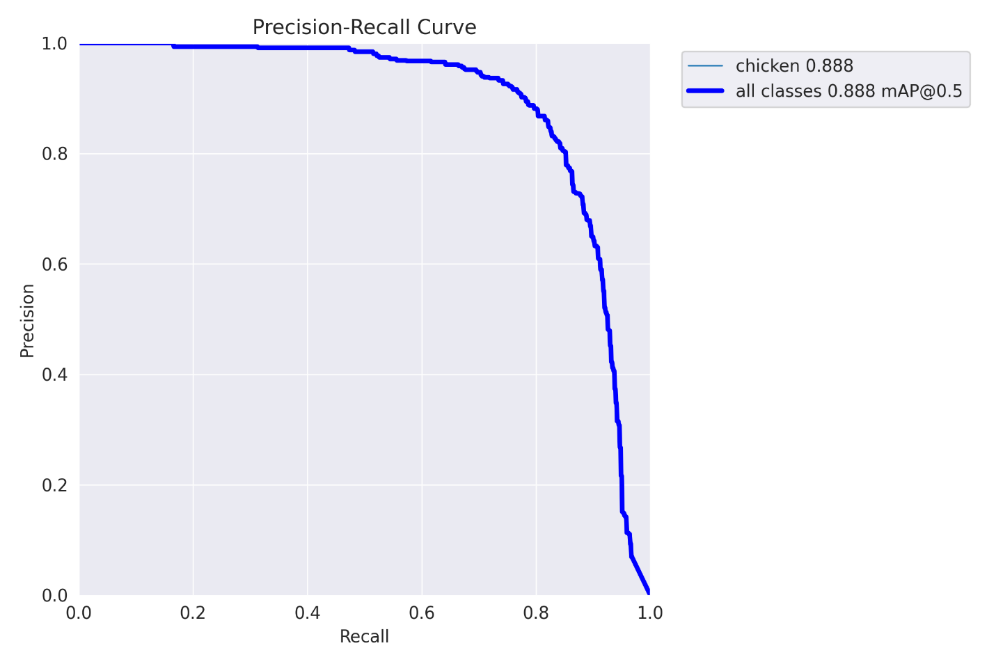
Đồ thị biểu diễn Precision, Recall theo Epoch trong quá trình Train



Đánh giá:

Nhìn chung, Precision và Recall có thể dao động do mô hình đang học. sau một số epochs, chúng ổn định hơn, cải thiện dần và dao động ở mức cao. Điều này là một dấu hiệu tốt cho thấy mô hình đang học hỏi và tối ưu hóa tốt hơn.

 Precision cao hơn Recall trong nhiều epochs cho thấy mô hình thận trọng hơn trong việc đưa ra các dự đoán dương tính.

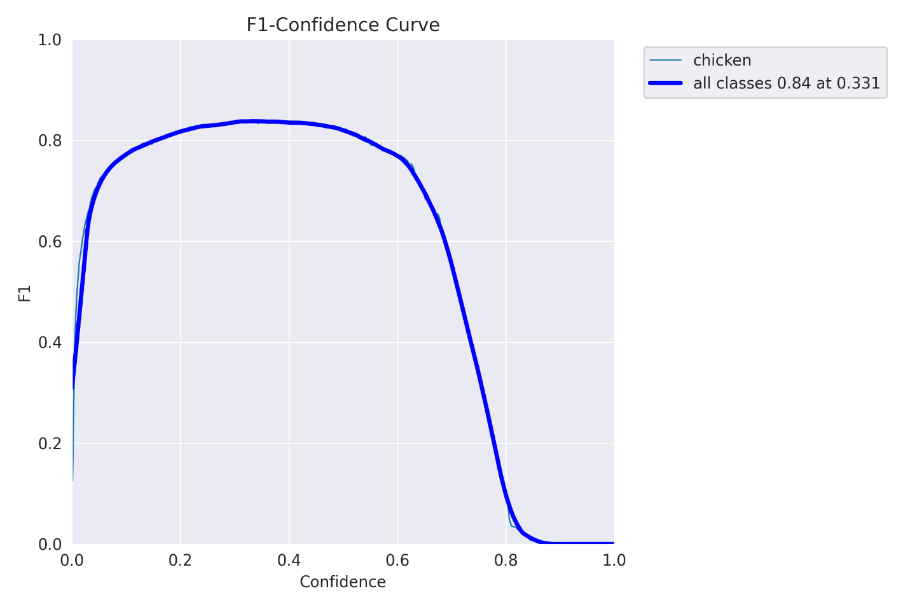
Đồ thị biểu diễn Precision-Recall Curve:

Hình 11: Đường cong Precison-Recall

**Nhận xét:**

Đường cong hướng về góc phải trên của biểu đồ, cho thấy cả Precison và Recall cao điều này cho thấy mô hình có hiệu suất tốt khi phân loại và phát hiện đối tượng

Đồ thị biểu diễn F1-score trong quá trình huấn luyện:



**Nhận xét:**

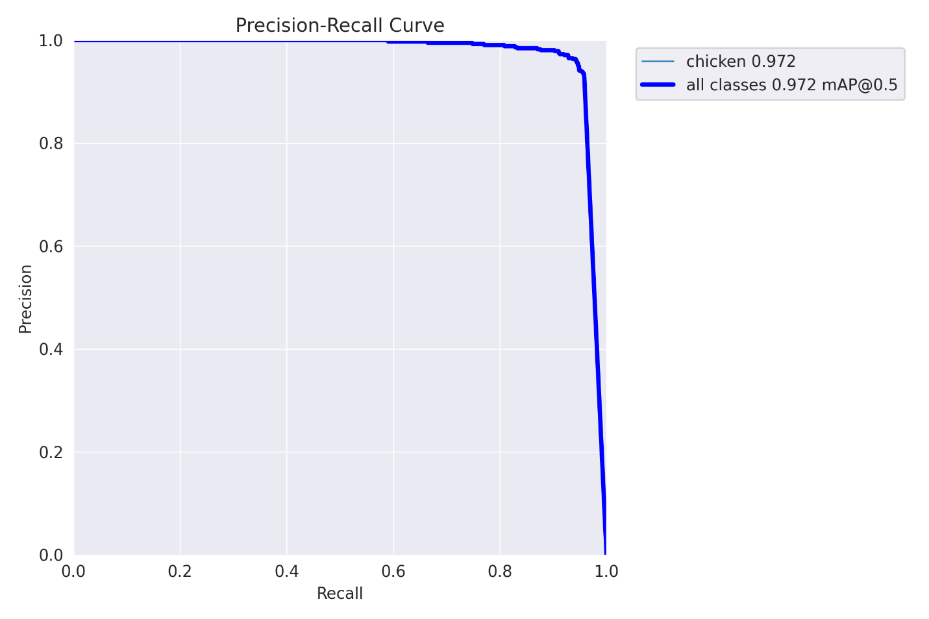
 **Điểm F1 cao nhất là 0.84** cho thấy mô hình có hiệu suất tốt trong việc cân bằng giữa Precision và Recall.

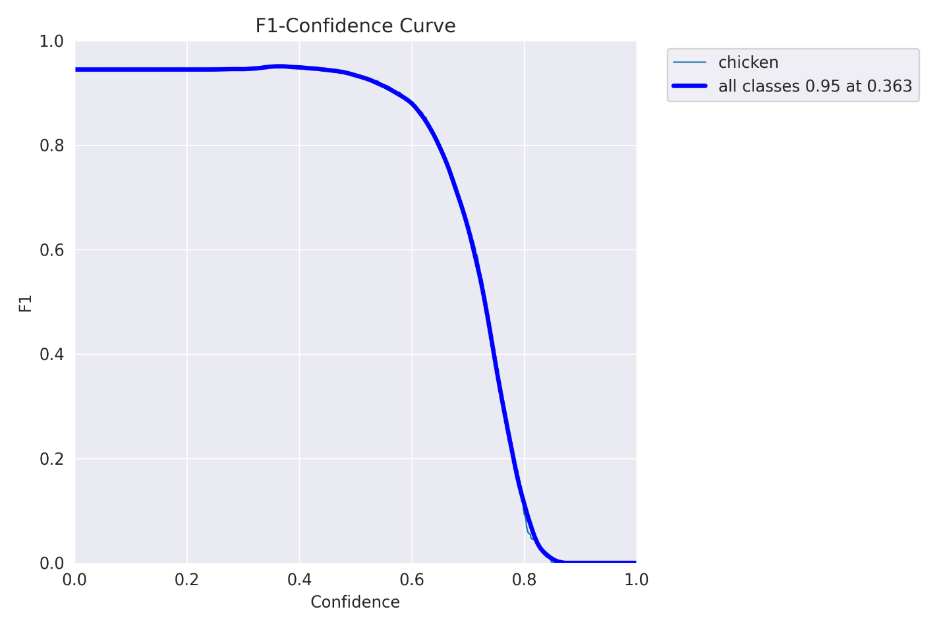
 **Ngưỡng tin cậy 0.331** cho thấy mô hình có xu hướng ưu tiên việc nhận diện nhiều mẫu dương tính hơn, chấp nhận một số lượng lớn các dự đoán sai dương tính để không bỏ sót quá nhiều mẫu dương tính thật sự.

**Điểm F1 cao nhất là 0.84 tại ngưỡng tin cậy 0.331** là một chỉ số rất tốt, cho thấy mô hình có hiệu suất mạnh mẽ và khả năng cân bằng giữa Precision và Recall.

**(1) So sánh, đánh giá hiệu suất của model trên tập kiểm tra**

Precision-Recall trên tập kiểm tra:



F1-Score trên tập kiểm tra:

Trên Tập Kiểm Tra (Test)

* **PR-curve (Precision-Recall curve): 0.972**
  + Điểm PR-curve là 0.972 là rất cao, cho thấy mô hình duy trì một sự cân bằng rất tốt giữa Precision và Recall trên tập kiểm tra. So với trước đây (0.922), điểm này cũng đã được cải thiện, cho thấy mô hình có khả năng tổng quát hóa rất tốt và hoạt động tốt hơn trên dữ liệu chưa từng thấy.
* **F1-score: 0.95**
  + Điểm F1 là 0.95 là xuất sắc, cho thấy mô hình hiện tại không chỉ duy trì mà còn cải thiện đáng kể hiệu suất so với trước đây (0.88). Mô hình có khả năng cân bằng tốt giữa Precision và Recall, và sự cải thiện này cho thấy mô hình rất mạnh mẽ trên tập kiểm tra.

### Đánh giá tổng quan về quán trình huấn luyện mô hình

Sau khi thực hiện 2 lần huấn luyện với 2 dataset khác nhau nhằm tăng hiệu suất nhóm đã có những đánh giá sau:

 **Cải Thiện Hiệu Suất Trên Tập Huấn Luyện**:

* Sự cải thiện từ PR-curve 0.859 lên 0.888 và F1-score từ 0.833 lên 0.84 cho thấy mô hình của bạn đã học tốt hơn từ dữ liệu huấn luyện. Điều này có thể do mô hình đã được tối ưu hóa tốt hơn hoặc dữ liệu huấn luyện đã được làm sạch và chuẩn bị tốt hơn.

 **Cải Thiện Hiệu Suất Trên Tập Kiểm Tra**:

* Sự tăng từ PR-curve 0.922 lên 0.972 và F1-score từ 0.88 lên 0.95 cho thấy mô hình không chỉ tổng quát hóa tốt mà còn rất mạnh mẽ trên dữ liệu mới. Đây là dấu hiệu tích cực, cho thấy mô hình có khả năng thực hiện tốt trên các dữ liệu thực tế chưa từng thấy.

 **Tổng Quan Hiệu Suất**:

* Với cả PR-curve và F1-score đều cao trên cả tập huấn luyện và tập kiểm tra, mô hình của bạn đang hoạt động rất tốt. Điều này cho thấy mô hình có độ tin cậy cao và ít có khả năng bị overfitting hoặc underfitting.

### Kết Luận

* **Hiệu Suất Mô Hình**: Mô hình hiện tại có hiệu suất rất tốt với cả PR-curve và F1-score cao trên cả tập huấn luyện và tập kiểm tra. Sự cải thiện này cho thấy mô hình đã được tối ưu hóa tốt hơn và có khả năng tổng quát hóa rất mạnh mẽ.
* **Khả Năng Áp Dụng**: Mô hình này có khả năng áp dụng vào thực tế cao, đặc biệt trong các tình huống yêu cầu sự cân bằng tốt giữa Precision và Recall.

### Hình ảnh nhận diện gà từ ESP32-CAM với model đã train.

## ****Giải pháp tự động cho gà ăn và cân khối lượng thức ăn, nước uống****

### A machine with a tube and a piece of paper Description automatically generated with medium confidenceTriển khai thực tế

**Hình 9: Hình ảnh thực tế module tự động cho ăn và cân khối lượng thức ăn**

**Hình**A plastic container and a box next to a printer

Description automatically generated **10: Hình ảnh thực tế module tự động cho gà ăn và cân khối lượng thức ăn**

### Kết quả

Chức năng cân thức ăn, nước uống hoạt động đúng theo thiết kế, thời gian trễ khi gửi dữ liệu lên server không quá 1 giây, độ chính xác cân đo khoảng 2g là trong mức sai số thiết kế của cân, hoàn toàn chấp nhận được đối với mục tiêu cân khối lượng thức ăn, nước uống.

**Hình 11: Kết quả chức năng cân khối lượng thức ăn**

A screenshot of a computer

Description automatically generatedChức năng tự động cho ăn hoạt động đúng theo thiết kế, độ trễ tín hiệu cho servo quay không quá 1 giây.

**Hình**A glass with a white object in it

Description automatically generated with medium confidence **12: Kết quả chức năng tự động cho ăn**

## Giải pháp nhận diện gà bệnh

### Dữ liệu

### Kết quả giải pháp nhận diện gà bệnh

Giải pháp

## Server

### API

**Images**

|  |  |
| --- | --- |
| **Tên chức năng** | Xóa hình ảnh khỏi cơ sở dữ liệu |
| **METHOD** | DELETE |
| **URL** | /image/delete |
| **Parameters** | **uuid**: Unique id của hình ảnh đó |

|  |  |
| --- | --- |
| **Tên chức năng** | Lấy thông tin hình ảnh |
| **METHOD** | GET |
| **URL** | /image/get |
| **Parameters** | **uuid**: Unique id của hình ảnh đó |

|  |  |
| --- | --- |
| **Tên chức năng** | Lấy thông tin hình ảnh mới nhất trên cơ sở dữ liệu |
| **METHOD** | GET |
| **URL** | /image/last |
| **Parameters** | None |

|  |  |
| --- | --- |
| **Tên chức năng** | Tìm kiếm các hình ảnh thỏa điều kiện |
| **METHOD** | GET |
| **URL** | /search |
| **Parameters** | **time**: Mốc thời gian hình ảnh cần tìm **from\_time**, **to\_time**: khoản thời gian chứa hình ảnh cần tìm  **minimum\_temp**: nhiệt độ tối thiểu được phát nhiện thông qua dữ liệu nhiệt độ đi kèm ảnh đó  **limit**: số lượng hình ảnh tối đa có thể trả về trong một request |

|  |  |
| --- | --- |
| **Tên chức năng** | Tải hình ảnh và dữ liệu lên CSDL |
| **METHOD** | POST |
| **URL** | /image/upload |
| **Parameters** | **file**: Hình ảnh được tải lên  **amg**: dữ liệu ảnh hồng ngoại |

**Sensors**

|  |  |
| --- | --- |
| **Tên chức năng** | Lấy dữ liệu của cảm biến đo được |
| **METHOD** | GET |
| **URL** | /sensor/ |
| **Parameters** | **time**: Mốc thời gian chứa dữ liệu cần tìm **from\_time**, **to\_time**: khoản thời gian chứa dữ liệu cần tìm |

|  |  |
| --- | --- |
| **Tên chức năng** | Tải lên dữ liệu cảm biến đo được |
| **METHOD** | POST |
| **URL** | /sensor/ |
| **Parameters** | **food\_weight**: Giá trị của cảm biến cân nặng dùng để cân lượng thức ăn  **water\_weight**: Giá trị của cảm biến cân nặng dùng để cân lượng nước |

**Analyse**

|  |  |
| --- | --- |
| **Tên chức năng** | Tính lượng thức ăn, nước uống tiêu thụ |
| **METHOD** | GET |
| **URL** | /analyze/consumed |
| **Parameters** | **from\_time**, **to\_time**: khoản thời gian chứa dữ liệu cần tính |

### Tốc độ thực thi hệ thống

|  |  |  |
| --- | --- | --- |
| **URL** | **Method** | **Thời gian (ms)** |
| /image/delete | DELETE | 118 |
| /image/get | GET | 125 |
| /image/last |
| /image/search | GET | 550 |
| /image/upload | POST | 2305I |
| /sensor/ | GET | 118 |
| /sensor/ | POST | 98 |
| /analyze/consumed | GET | 210 |

*I) Thời gian thực hiện của AI: 2244ms*

#### Bảng thời gian thực thi API trước (v1) và sau khi được tối ưu (v2), method: /image/search

|  |  |  |
| --- | --- | --- |
| **Số lượng hình ảnh yêu cầu xử lý** | **Thời gian thực thi (ms)** | |
| **APIv1** | **APIv2** |
| 100 | 2722 | 441 |
| 200 | 4240 | 550 |
| 300 | 6318 | 711 |
| 400 | 7842 | 896 |
| 500 | 10079 | 1054 |
| 600 | 12463 | 1226 |
| 700 | 13928 | 1376 |
| 800 | 15749 | 1719 |
| 900 | 18054 | 1898 |
| 1000 | 19386 | 1964 |
| 1100 | 21837 | 2073 |
| 1200 | 23429 | 2210 |

A graph with a line

Description automatically generated

**Hình 13: Biểu đồ tương quan thời gian thực hiện của API v1 cũ và v2 đã được tối ưu.**

## Ứng dụng di động

### Thiết kế

Ứng dụng có 3 chức năng chính ở đặt ở cuối màn hình bao gồm:

* Lịch để xem gà theo ngày cụ thể
* Biểu đồ để xem lượng thức ăn hoặc nước / thời gian theo ngày cụ thể
* Thông báo để xem gà có nhiệt độ cao hơn mức nhiệt độ trung bình

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

### Kết quả

* Hiển thị bảng tổng quan và danh sách ảnh gà theo ngày, có thể chọn để xem chi tiết
* Hiển thị thông báo trong ứng dụng nếu nhiệt độ gà cao hơn nhiệt độ trung bình
* Hiển thị biểu đồ lượng thức ăn và nước theo thời gian
* Có thể sử dụng App mà không cần Internet, tuy nhiên không hiển thị được biểu đồ do thư viện AnyChart cần quyền truy cập Internet để hiển thị biểu đồ
* Cải thiện hiệu suất trải nghiệm người dùng
* Tốc độ thực thi ứng dụng phụ thuộc vào lượng dữ liệu của server

A screenshot of a bird

Description automatically generatedA screenshot of a graph

Description automatically generated

# Kết luận

## Đánh giá

Sau khi hoàn thiện đồ án nhóm đã có những đánh giá về đồ án như sau

* Đã xây dựng được phần cứng với các chức năng đã đề ra, các cảm biến hoạt động độc lập và có khả năng
* Đã có hệ thống giao tiếp giữa phần cứng và server, server tiếp nhận và xử lí dữ liệu và xây dựng thành các API để ứng dụng di động có thể sử dụng
* Đã xây dựng model nhận diện gà và đếm được số lượng gà trong chuồng từ ảnh mà ESP32-CAM chụp và gửi về hệ thống
* Có thể xác định được nhiệt độ của từng con gà thông qua AMG8833
* Xây dựng được ứng dụng để có thể xem được thông tin về số lượng gà trong chuồng, lượng thức ăn tiêu thụ và đưa ra dự đoán nếu có gà bị bệnh.

Nhưng bênh cạnh đó vẫn còn những điểm chưa được ưng ý với sản phẩm:

* Nhóm đã tạo ra được ứng dụng để có thể xem được số liệu thu thập từ chuồng gà, nhưng chỉ dừng lại ở mức độ lấy dữ liệu từ server và hiện lên, chưa có nhiều chức năng CRUD để phục vụ người dùng nên ứng dụng gây ra sự khó chịu khi sử dụng
* Chưa xây dựng được cơ sở dữ liệu để lưu trữ thông tin lấy từ API.

## Hướng phát triển

**Nhận diện và đếm gà**: Tuy mô hình đã có thể nhận diện và đếm gà 1 cách chính xác và đúng nhưng nhóm cũng muốn thử sức với 1 camera có độ phân giải tốt hơn để có thể nhận diện tốt, nhanh với độ chính xác mà model đã xây dựng.

**Xác định nhiệt độ gà:** Giải pháp nhận diện nhiệt độ gà qua AMG8833 của nhóm là đáp ứng yêu cầu đề ra để giải quyết vấn đề của đề tài nhưng nhóm gặp khó khăn vì camera hồng ngoại của nhóm không thực sự tốt để đo nhiệt độ đúng với độ xấp xỉ nhỏ hơn

**Ứng dụng di động:** Nhóm sẽ đầu tư hơn về mặt giao diện và xây dựng 1 hệ thống lưu trữ dữ liệu trên máy, tạo ra thêm các chức năng CRUD và sắp xếp lại luồng logic để người dùng dễ dàng sử dụng ứng dụng

# Danh mục tài liệu tham khảo