ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

MÔ HÌNH ĐÁNH GIÁ CHỈ TIÊU CHẤT LƯỢNG SẢN XUẤT SỢI

Người hướng dẫn: **TS. NGUYỄN HOÀNG MAI**

Sinh viên thực hiện:

1. LÊ ĐỨC THỊNH – MSSV: 105200387 – LỚP: 20TDHCLC1

2. PHẠM LÊ QUANG HUY – MSSV: 105200452 – LỚP: 20TDHCLC1

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài : Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi

Sinh viên thực hiện : Lê Đức Thịnh

Phạm Lê Quang Huy

Số thẻ SV : 105200387

105200452

Lớp : 20TDHCLC1

Nội dung đề tài : Xây dựng hệ thống sử dụng trí tuệ nhân tạo AI và điện toán đám mây Cloud để phát hiện lỗi côn sợi (tách xơ ngoại lai, sai thành phần) một cách tự động, chính xác và theo thời gian thực, giúp thay thế phương pháp thủ công và tăng hiệu quả khi dễ dàng xử lý lỗi từ xa.

Mô hình huấn luyện dựa theo cấu trúc YoLov8n được thiết kế để thu thập các lỗi có được từ các công ty gửi về, dữ liệu học được sẽ đưa lên Cloud AWS để người dùng ở bất cứ đâu có thể truy cập và gửi ảnh nhận diện. Kết quả ảnh nhận diện sẽ được trả lại cho người dùng và được lưu trữ trên Cloud với các mốc thời gian để thuận tiện cho việc theo dõi.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
Lê Đức Thịnh	105200387	20TDHCLC1	Kỹ thuật điều khiển & Tự động hóa
Phạm Lê Quang Huy	105200452	20TDHCLC1	Kỹ thuật điều khiển & Tự động hóa

1. *Tên đề tài đồ án:*
   Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi
2. *Đề tài thuộc diện:* ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
4. *Nội dung các phần thuyết minh và tính toán:*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Lê Đức Thịnh	- Xây dựng mô hình để huấn luyện - Chuẩn bị dữ liệu để huấn luyện - Xây dựng cơ sở lý thuyết - Phân tích cách học của mô hình YOLOv8n - Kiểm tra thực nghiệm và đánh giá mô hình - Viết thuyết minh báo cáo
2	Phạm Lê Quang Huy	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Lê Đức Thịnh	- Xây dựng và thiết kế Cloud AWS
2	Phạm Lê Quang Huy	- Nghiên cứu và phân tích thuật toán CNN

5. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Hoàng Mai	- Giao đề tài - Định hướng chi tiết về đề tài - Hướng dẫn và gợi ý cặn kẽ cho mô hình - Sửa chữa thuyết minh

6. *Ngày giao nhiệm vụ đồ án:* *26/ 02/ 2025.*
7. *Ngày hoàn thành đồ án:* *07/ 06/ 2025*

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
Lê Đức Thịnh	105200387	20TDHCLC1	Kỹ thuật điều khiển & Tự động hóa
Phạm Lê Quang Huy	105200452	20TDHCLC1	Kỹ thuật điều khiển & Tự động hóa

Tên đề tài ĐATN:

Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi

Họ tên người HD: TS. Nguyễn Hoàng Mai

Đơn vị: Đại học Bách Khoa ĐN

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		Đã thực hiện (%)	**Tiếp tục thực hiện (%)**	
1	24/2-2/3	-	-	
2	3/3-9/3	-	-	
3	10/3-16/3	-	-	
4	17/3-23/3	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ %: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	24/3-30/3	-	-	
6	31/3-6/4	-	-	
7	7/4-13/4	-	-	
8	14/4-20/4	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ %: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		

9	21/4-27/4	-	-	
10	28/4-4/5	-	-	
11	5/5-11/5	-	-	
12	12/5-18/5	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	19/5-25/5	-	-	
14	26/5-1/6	-	- .	
15	2/6-8/6	- Hoàn thành đồ án	-	

LỜI NÓI ĐẦU

Trong suốt khoảng thời gian học tại trường Đại Học Bách Khoa – Đại Học Đà Nẵng cũng như trong thời gian thực hiện và hoàn thành đồ án này, nhóm em đã nhận được sự giảng dạy và hỗ trợ tận tình từ quý Thầy, Cô trong khoa Điện cùng một số Thầy, Cô thuộc các khoa khác. Nhóm em xin chân thành cảm ơn đến quý Thầy, Cô không chỉ truyền đạt kiến thức chuyên môn mà còn định hướng cho nhóm em tinh thần học tập nghiêm túc để có được nền tảng quý giá giúp em hoàn thành đồ án tốt nghiệp này một cách hiệu quả.

Đặc biệt đề tài lần này là kết quả của quá trình học tập, nghiên cứu nghiêm túc dưới sự hướng dẫn tận tình và chu đáo của Thầy **TS.Nguyễn Hoàng Mai**. Nhóm xin chân thành cảm ơn thầy đã luôn hỗ trợ, định hướng, thường xuyên có những buổi gặp mặt để thảo luận, giải đáp thắc mắc và tạo điều kiện thuận lợi để nhóm hoàn thành đề tài.

Tiếp theo nhóm em xin gửi lời cảm ơn Công Ty Cổ Phần Sợi Hoà Thọ và các anh chị trong công ty đã hỗ trợ, tạo điều kiện truyền đạt những kiến thức, kinh nghiệm thực tiễn giúp nhóm em tiếp cận được với môi trường làm việc thực tế, bổ trợ cho quá trình thực hiện đồ án tốt nghiệp lần này.

Trong quá trình thực hiện, tuy đã nổ lực hết mình, nhưng còn hạn chế thời gian và thực tiễn, đề tài của nhóm em vẫn không tránh khỏi được những thiếu sót. Nhóm rất mong nhận được sử góp ý của thầy cô và hội đồng để hoàn thiện hơn trong các nghiên cứu và ứng dụng thực tế sau này.

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Nhóm em xin cam đoan rằng đồ án tốt nghiệp với đề tài : "Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi" là kết quả của quá trình học tập, nghiên cứu và độc lập của bản thân dưới sự hướng dẫn của thầy **Nguyễn Hoàng Mai**.

Trong quá trình thực hiện đồ án, nhóm em đã tuân thủ đầy đủ các quy định về đạo đức học thuật và không sao chép, sử dụng trái phép bất kỳ nội dung nào từ các nguồn tài liệu khác. Mọi trích dẫn tham khảo trong báo cáo đều được ghi rõ nguồn gốc.

Nhóm em hoàn toàn chịu trách nhiệm trước nhà trường và pháp luật về tính trung thực và liêm chính học thuật của đồ án này.

Sinh viên thực hiện

MỤC LỤC

DANH SÁCH CÁC BẢNG VÀ HÌNH VẼ

MỞ ĐẦU

1. Mục đích thực hiện đề tài :

Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, việc ứng dụng các công nghệ mới như trí tuệ nhân tạo (AI), thị giác máy tính và điện toán đám mây (Cloud Computing) vào sản xuất ngày càng được quan tâm. Ngành công nghiệp dệt may, vốn là ngành mũi nhọn của Việt Nam, chất lượng của côn sợi đóng vai trò quan trọng trong việc quyết định hiệu suất và giá trị của sản phẩm đầu ra.

Tuy nhiên, trên thực tế, việc kiểm tra và phát hiện lỗi trên côn sợi hiện nay chủ yếu vẫn được thực hiện bằng phương pháp thủ công. Điều này không chỉ tốn thời gian, chi phí nhân công mà còn thiếu độ ổn định và chính xác. Từ nhu cầu cấp thiết trên, nhóm em xin thực hiện đề tài : "Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi". Việc xây dựng một hệ thống chẩn đoán kỹ thuật tự động, ứng dụng công nghệ học sâu và tích hợp lên nền tảng điện toán đám mây sẽ giúp cải thiện đáng kể hiệu suất, độ tin cậy và khả năng nhận diện từ xa trong các nhà máy sản xuất sợi.

2. Mục tiêu của đề tài :

- Xây dựng mô hình nhận diện lỗi côn sợi sử dụng mạng học sâu YOLOv8n.
- Tích hợp mô hình vào websever để người dùng có thể tải ảnh lên và nhận diện kết quả, phân tích lỗi trực tuyến.
- Triển khai toàn bộ hệ thống trên nền tảng AWS, sử dụng các dịch vụ như EC2, S3, DynamoDB.
- Hỗ trợ khả năng cập nhật mô hình theo thời gian, mở rộng nhận diện thêm các lỗi mới.
- Tối ưu hoá hiệu năng và chi phí vận hành của hệ thống trên môi trường cloud thực tế.

3. Phạm vi và đối tượng nghiên cứu :

- Đối tượng nghiên cứu : Các lỗi thường gặp trên bề mặt côn sợi như : Tách Xơ Ngoại Lai (lẫn tạp chất, sợi khác màu), Sai Thành Phần (tỷ lệ sợi không đúng).
- Phạm vi nghiên cứu :
  - Sử dụng mô hình YOLOv8n để huấn luyện và phát hiện lỗi trong ảnh chụp từ côn sợi.
  - Phát triển giao diện người dùng websever.
  - Triển khai hệ thống xử lý và lưu trữ trên nền tảng AWS.

4. Phương pháp nghiên cứu :

- Xây dựng mô hình và giao diện websever để các gửi dữ liệu huấn luyện.

- Thu thập, xử lý và gắn nhãn dữ liệu ảnh côn sợi.
- Huấn luyện mô hình nhận diện sử dụng YOLOv8n.
- Xử dụng xử lý ảnh để tối ưu đầu vào.
- Xây dựng giao diện websever cho phép tải ảnh và hiển thị kết quả.
- Triển khai hệ thống AI lên AWS EC2, lưu trữ dữ liệu trên S3 và quản lý lịch sử bằng DynamDB.
- Đánh giá kết quả nhận diện dựa trên độ chính xác và tính ứng dụng thực tiễn.

5. Cấu trúc của đồ án tốt nghiệp :

Thuyết minh của đồ án được trình gồm các chương sau :

Chương 1 : Tổng quan về đề tài – Nêu lên thực trạng, vai trò của AI trong sản xuất, và nhu cầu tự động hoá kiểm tra lỗi côn sợi.

Chương 2 : Xây dựng mô hình và cơ sở lý thuyết – Trình bày lý thuyết về xử lý ảnh, thị giác máy tính, CNN, YOLO, điện toán đám mây. Xây dựng sơ đồ và cấu trúc huấn luyện cho đề tài.

Chương 3 : Thiết kế cụ thể cho mô hình – Phân tích chi tiết cấu trúc YOLOv8n, từng bước xây dựng chi tiết mô hình và triển khai Cloud.

Chương 4 : Kết quả và đánh giá – Trình bày kết quả nhận diện lỗi, giao diện hệ thống, đánh giá độ chính xác và hiệu quả.

Kết luận và hướng phát triển – Tổng kết kết quả đạt được và đề xuất cải tiến trong tương lai.

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Thực trạng hiện này :

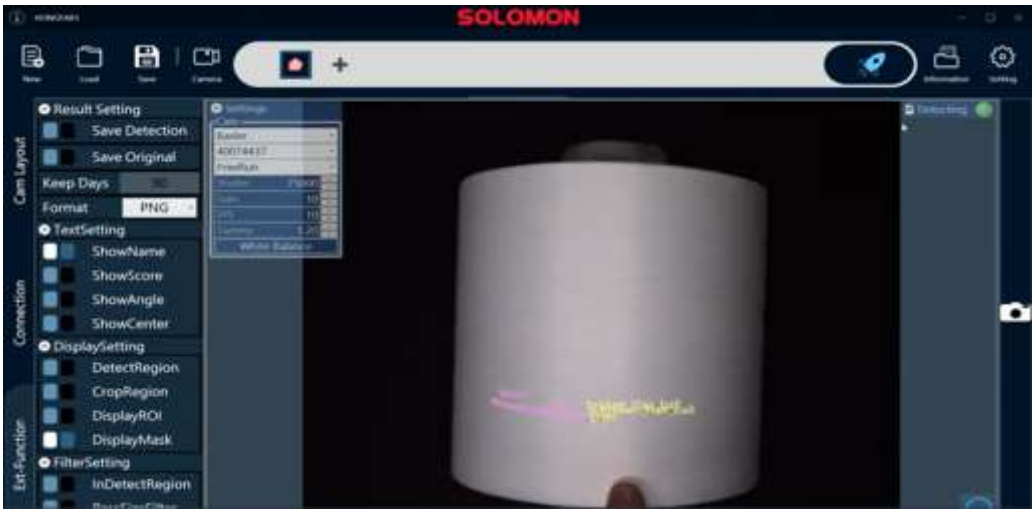
Sau khi khảo sát thực tiến và tìm hiểu trên các trang mạng xã hội thì nhóm em thấy rằng, các côn sợi khi thành phẩm thường phát sinh một số lỗi như:

- Sai thành phần (do trộn nhầm nguyên liệu hoặc điều chỉnh sai thông số)
- Sợi bẩn, sợi dơ
- Có các vật lạ lẫn vào sợi
- …

Hình 1. 1 Một số lỗi phổ biến

Các quốc gia và doanh nghiệp tiên phong trong lĩnh vực này gồm có:

- Trung Quốc dẫn đầu trong hiện đại hóa ngành dệt may bằng cách ứng dụng AI thị giác máy để kiểm tra lỗi sợi, dị vật và màu sắc. Dữ liệu lỗi được đồng bộ lên cloud, giúp theo dõi sản xuất thời gian thực và phân tích xu hướng để cải tiến quy trình.
- Công ty Solomon 3D tại Đài Loan phát triển hệ thống SolVision sử dụng AI để kiểm tra chất lượng sợi. Hệ thống phát hiện lỗi bề mặt qua phân tích hình ảnh, giúp tăng độ chính xác và giảm thời gian kiểm tra so với phương pháp thủ công.



Hình 1. 2 Phát hiện lỗi sợi

Đối với nước ta:

- Việt Nam là một trong những quốc gia xuất khẩu sợi lớn, đặc biệt sang Trung Quốc, Hàn Quốc, Thổ Nhĩ Kỳ và Ấn Độ. Tuy nhiên, tỷ lệ gia công đơn thuần còn cao, giá trị gia tăng thấp vì đa số doanh nghiệp chỉ dừng lại ở khâu kéo sợi, không chủ động được khâu dệt-nhuộm-hoàn tất.
- Công nghệ sản xuất và kiểm tra của nước ta hiện nay:
  - Về khâu kiểm tra chất lượng sản phẩm (côn sợi):
    - Chủ yếu vẫn kiểm tra thủ công: công nhân dùng mắt thường để kiểm tra lỗi ngoại quan như sợi không đều, có tạp chất,…
- Về ứng dụng AI và điện toàn đám mây:
  - Tính đến năm 2024, chua có ghi nhận rộng rãi về việc doanh nghiệp sợi tại Việt Nam áp dụng AI để kiểm tra lỗi của côn sợi.
  - Lý do:
    - Chi phí đầu tư cho AI và thị giác máy tính còn cao.
    - Thiếu nhân lực có chuyên môn về trí tuệ nhân tạo (AI) trong ngành sợi.
    - Doanh nghiệp còn thận trọng trong thay đổi công nghệ, đặc biệt là các doanh nghiệp vừa và nhỏ.
- Từ những vấn đề đã nêu trên tạo cho chúng ta những thách thức và tồn tại :

Bảng 1 1 Một số vấn đề còn tồn tại ở Việt Nam

Vấn đề	Mô tả
Thiếu tự động hóa và kiểm tra lại	Dễ sai sót, thiếu nhất quán
Chất lượng không ổn định	Phụ thuộc vào kỹ năng người kiểm tra
Tốn nhiều thời gian	Phương pháp kiểm tra thủ công tốn nhiều thời gian
Không có dữ liệu theo dõi	Không lưu trữ được dữ liệu lỗi để phân tích và cải tiến quá trình

- Cơ hội chuyển đổi:
  - Với sự phát triển của các mô hình AI "nhẹ" và rẻ hơn, các doanh nghiệp có thể:
    - Sử dụng camera công nghiệp kết hợp với thuật toán học máy đơn giản.
    - Lưu trữ dữ liệu lỗi qua điện toán đám mây để phân tích lỗi theo thời gian.

1.2. Nội dung thực hiện :

1.2.1. Giải pháp để xử lý các sản phẩm (côn sợi) lỗi ở cuối quy trình sản xuất:

Từ những vấn đề đã nêu về thực trạng của ngành công nghiệp sợi ở nước ta hiện nay, nhóm em xin đưa ra giải pháp cũng là đề tài của dự án này đó là "Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi". Với đề tài này, nhóm em sẽ thực hiện huấn luyện một mô hình (sử dụng phiên bản YOLOv8n) để nhận diện một số lỗi của côn sợi (cụ thể là lỗi sai thành phần và lỗi tách sơ ngoại lai). Sau khi huấn luyện xong, mô hình sẽ được tích hợp lên nền tảng cloud để có thể xử lý tự động các hình ảnh lỗi được gửi lên theo thời gian thực. Dữ liệu này sẽ được lưu trữ có hệ thống nhằm phục vụ việc phân tích, theo dõi xu hướng lỗi theo thời gian.

Giải pháp trên mang lại các ưu điểm cũng như một số thách thức như:

- Ưu điểm:
  - Hệ thống cho phép phát hiện lỗi tự động và theo thời gian thực, giúp giảm thời gian kiểm tra thủ công.
  - Dữ liệu xử lý và lưu trữ trên cloud, hỗ trợ truy cập từ xa và phân tích.
  - Việc lưu ảnh theo thời gian giúp theo dõi nguyên nhân và xu hướng lỗi hiệu quả.
- Nhược điểm:

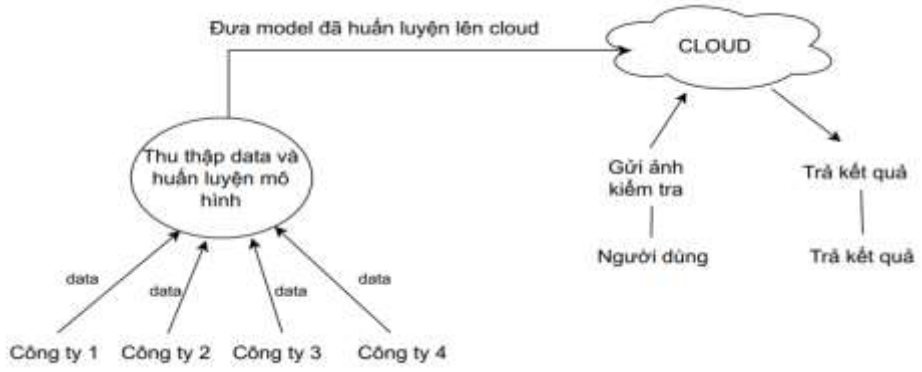
- Phụ thuộc vào chất lượng dữ liệu: Dữ liệu không đa dạng hoặc bị nhiễu dẫn đến mô hình dễ học sai.
- Độ trễ: Việc gửi hình ảnh và chờ kết quả có thể gây trễ nếu hạ tầng mạng không ổn định hoặc dịch vụ Cloud chưa tối ưu.
- Chi phí Cloud: Tốn chi phí duy trì server, lưu trữ và xử lý dữ liệu nhất là dữ liệu hình ảnh nhiều và liên tục.

1.2.2. Sơ đồ tổng quan về giải pháp :

Hệ thống nhận diện lỗi sợi tự động được xây dựng theo mô hình client-sever, trong đó các công ty sẽ gửi dữ liệu ảnh về máy chủ đề tiến hành huấn luyện mô hình AI hoặc nhận diện lỗi. Dữ liệu và kết quả được quản lý, lưu trữ thông qua Cloud.

Mô tả luồng xử lý :

- Công ty gửi dữ liệu (ảnh và nhãn).
- Sever tiếp nhận dữ liệu và huấn luyện mô hình.
- Mô hình AI được cập nhật và triển khai lên nền tảng Cloud.
- Người dùng gửi ảnh lỗi cần nhận diện lên Cloud.
- Cloud xử lý ảnh bằng mô hình đã huấn luyện.
- Kết quả nhận diện được trả về người dùng và lưu trữ trên cloud.



Hình 1. 3 Sơ đồ tổng quan giải pháp

1.3. Các lỗi trong đề tài :

1.3.1. Lỗi tách xơ ngoại lai :

Lỗi tách xơ ngoại lai trong sản xuất sợi xảy ra khi các xơ khác loại hoặc tạp chất xơ không mong muốn bị lẫn vào dòng sợi chính, tạo ra sự không đồng đều về màu sắc, tính chất hoặc cấu trúc của sợi.

Hình 1. 4 Lỗi tách xơ ngoại lai

❖ Nguyên nhân :

- Nguyên liệu đầu vào không đồng nhất : Các loại xơ tự nhiên (bông, len...) hoặc xơ nhân tạo (polyester, viscose...) trộn lẫn không đúng tỷ lệ hoặc sai chủng loại.
- Lỗi trong quá trình mở và trộn xơ : Máy mở kiện hoặc máy trộn xơ không sạch sẽ, còn sót xơ từ mẻ trước.
- Môi trường sản xuất không sạch : Gió, bụi, hoặc các vật thể lạ như xơ vụn, tóc, sợi từ trang phục công nhân có thể rơi vào dòng xơ đang xử lý.

1.3.2. Lỗi sai thành phần :

Lỗi sai thành phần trong sản xuất sợi xảy ra khi tỷ lệ hoặc loại xơ sử dụng không đúng như yêu cầu kỹ thuật hoặc công thức phối trộn, dẫn đến sản phẩm không đạt tiêu chuẩn về tính chất cơ học, hóa học hoặc thẩm mỹ.

Hình 1. 5 Lỗi sai thành phần

❖ Nguyên nhân :

- Sai sót trong pha trộn xơ : Tỷ lệ pha trộn không đúng do máy trộn không được hiệu chuẩn chính xác hoặc bị hỏng.
- Quá trình nhập số liệu : Công nhân sai sót trong quá trình nhập số liệu vào máy móc, thiếu đào tạo hoặc không tuân thủ quy trình chuẩn.

1.4. Kết quả dự kiến :

Đối với đề tài này, nhóm em mong muốn đạt được các kết quả như sau:

- Huấn luyện thành công một mô hình có thể nhận diện được chính xác lỗi sai thành phần và tách sơ ngoại lai.
- Khi có dữ liệu lai, kết hợp giữa hai loại lỗi sai thành phần và tách sơ ngoại lai mô hình vẫn nhận diện được và đưa ra kết quả mong muốn.
- Tích hợp mô hình với Cloud.
- Sau khi tích hợp mô hình với Cloud thành công thực hiện nhận dữ liệu được gửi đến đưa ra kết quả chính xác cho khách hàng và lưu trữ dữ liệu đó.
- Mong muốn nhận diện lỗi một cách tự động để nâng cao sự chính xác, phát hiện sản phẩm lỗi một cách kịp thời để xử lý và giảm tối đa chỉ phí khắc phục.
- Khi nhận được dữ liệu mới hoàn toàn, mô hình thực hiện báo về cho admin gán nhãn, thực hiện huấn luyện với dữ liệu mới và vẫn giữ nhận dạng được các dữ liệu cũ.
- Tối ưu chi phí cho dự án.

1.5. Kết luận chương :

Ở chương 1, nhóm em đã nêu tổng quan về đề tài "Xây dựng mô hình đám mây chẩn đoán kỹ thuật cho các nhà máy công nghiệp sợi". Nhóm đưa ra thực trạng của các nhà máy công nghiệp sợi ở nước ta từ đó cho thấy đề tài lần này là vô cùng cần thiết để đem trí tuệ nhân tạo (AI) vào phục vụ con người. Cung cấp giải pháp và đưa ra kết quả mong muốn.

Chương 2 nhóm sẽ đi sâu vào cấu trúc, nguyên lý hoạt động, nêu ra các đề xuất cho đối tượng chính của đề tài lần này.

CHƯƠNG 2 : XÂY DỰNG MÔ HÌNH VÀ CƠ SỞ LÝ THUYẾT

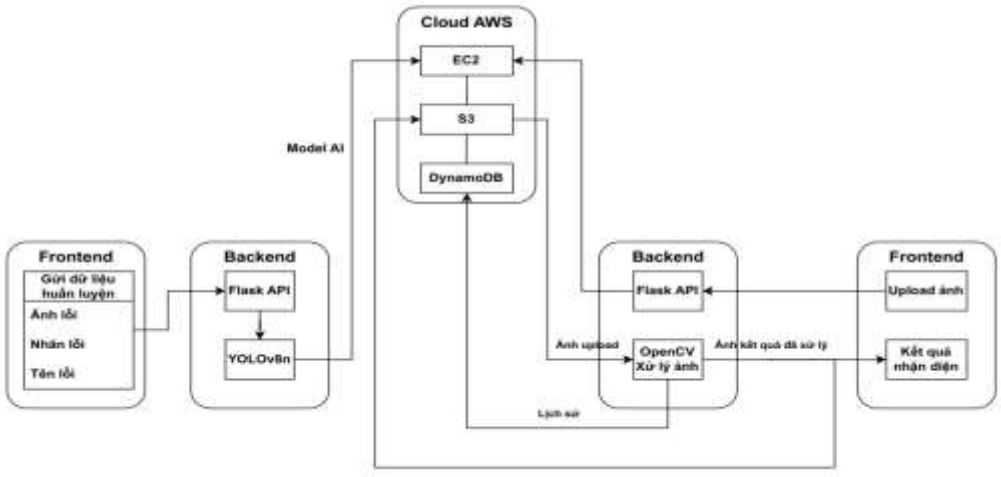
2.1. Cấu trúc :

2.1.1. Sơ đồ khối :

Hệ thống được chia thành các thành phần chi tiết như sau :

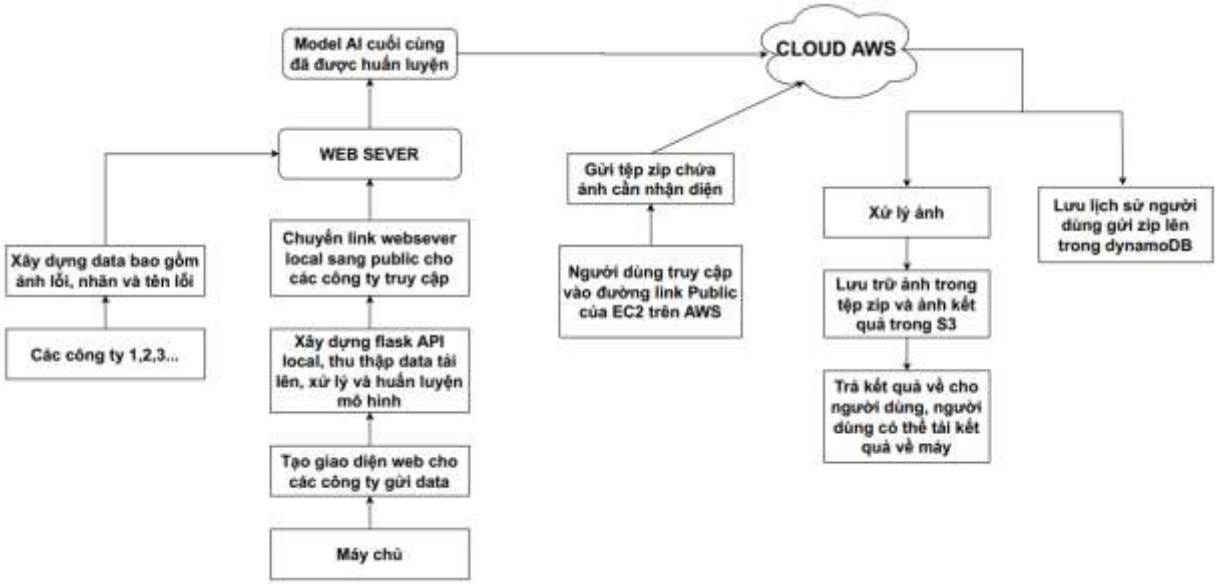
- **Frontend** : Giao diện web để người dùng upload dữ liệu huấn luyện hoặc ảnh cần nhận diện lỗi. Giao diện này được thiết kế đơn giản, trực quan và dễ sử dụng với các chức năng chính như tải ảnh, xem kết quả và tải về kết quả.
- **Backend** : Tập hợp các thành phần chạy phía máy chủ, chịu trách nhiệm xử lý các yêu cầu từ frontend (giao diện người dùng) và cung cấp dữ liệu đã xử lý hoặc thông tin phản hồi. Trong đề tài của bạn, backend gồm:
  - Flask API : Nền tảng chính của backend, nó nhận ảnh hoặc dữ liệu từ frontend, gọi mô hình YOLOv8 để xử lý ảnh và nhận diện lỗi, trả kết quả về cho frontend, gửi và nhận dữ liệu nhận được từ cloud AWS.
  - Xử lý ảnh (OpenCV) : Một phần của backend để xử lý ảnh đầu vào, vẽ bounding box, định dạng kết quả.
- **Cloud AWS** :

  - EC2: Chạy Flask server và đảm nhiệm xử lý chính.
  - S3: Lưu trữ ảnh gốc và ảnh đã xử lý với cấu trúc thư mục rõ ràng.
  - DynamoDB: Lưu lịch sử truy cập và kết quả nhận diện theo thời gian, tên ảnh.



Hình 2. 1 Sơ đồ khối toàn hệ thống

2.1.2. Sơ đồ chi tiết :



Hình 2. 2 Sơ đồ chi tiết toàn hệ thống

2.2. Nguyên lý hoạt động của hệ thống :

2.2.1. Giai đoạn huấn luyện mô hình nhận diện lỗi sợi :

- Máy chủ tạo flask sever để nhận dữ liệu.
- Sử dụng ngrok để public đường link sever lên internet.
- Công ty gửi bộ dữ liệu gồm ảnh và nhãn định dạng YOLO.
- Hệ thống tự động tiếp nhận, kiểm tra và phân chia dữ liệu train/test.
- Mô hình YOLOv8 được huấn luyện trên dữ liệu gộp lại mỗi lần gửi lên dữ liệu mới, lưu lại lại mô hình sau mỗi lần huấn luyện thành công.
- Sau khi có mô hình cuối cùng thì tiến hành deploy lên Cloud để người dùng hoặc các doanh nghiệp gửi ảnh đến nhận diện.

2.2.2. Giai đoạn xử lý ảnh trên Cloud AWS :

- Flask sever trên EC2 nhận ảnh zip từ người dùng thông qua giao diện web.
- Server tiến hành giải nén file zip và đưa từng ảnh vào mô hình YOLOv8 để nhận diện lỗi.
- Kết quả nhận diện (bounding box, loại lỗi) được ghi vào ảnh bằng OpenCV.
- Ảnh kết quả sẽ được lưu vào thư mục riêng trong S3.
- Mỗi lần xử lý, server cập nhật thông tin ảnh đã xử lý (thời gian, tên lỗi, tên người gửi) vào DynamoDB.
- Trả ảnh kết quả và thông tin nhận diện lại cho người dùng qua frontend dưới dạng hình ảnh và dữ liệu JSON.

2.3. Xử lý ảnh và thị giác máy tính :

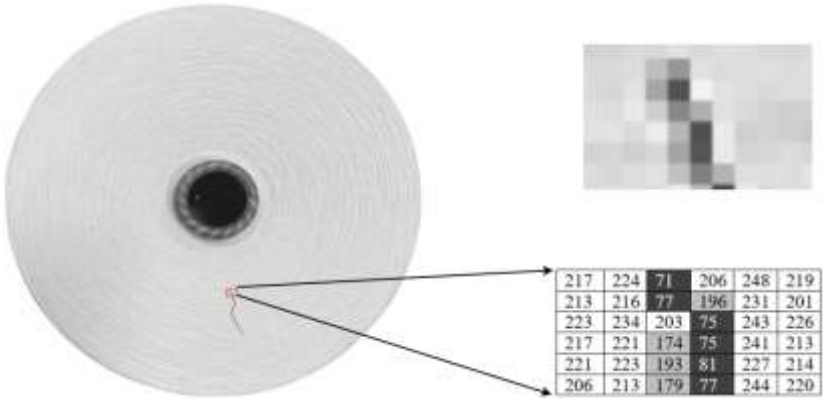
2.3.1. Tổng quan về xử lý ảnh :

a. Ảnh số và điểm ảnh :

❖ Điểm ảnh :

Là đơn vị cơ bản nhất để tạo nên một bước ảnh kỹ thuật số. Địa chỉ của điểm ảnh được xem như là một toạ độ (x,y) nào đó. Một bức ảnh kỹ thuật số có thể tạo ra bằng cách chụp hoặc bằng một phương pháp đồ hoạ nào khác, được toạ nên hàng ngàn hoặc hàng triệu pixel riêng lẻ. Bức ảnh chứa càng nhiều pixel thì càng chi tiết.

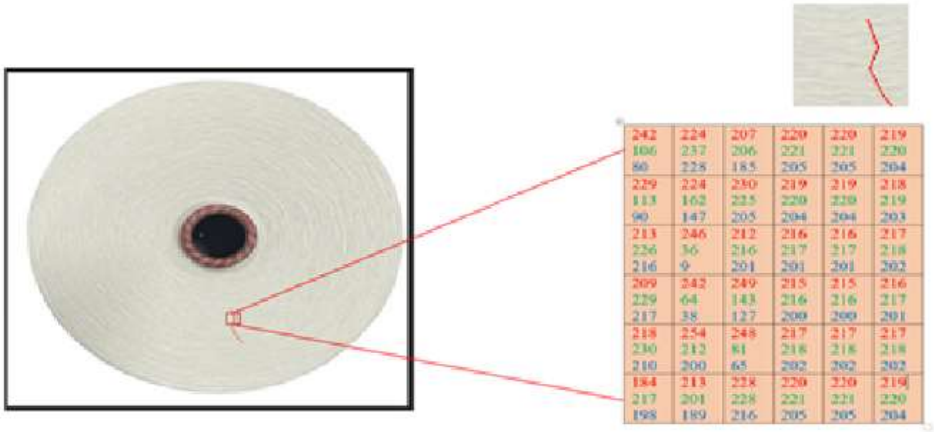
- Với ảnh trắng đen :



Hình 2. 3 Phân bố pixel trên ảnh xám

- Ảnh cuộn sợi được chuyển sang trắng đen.
- Mỗi điểm ảnh trong ảnh trắng đen sẽ có 1 giá trị với cường độ từ 0 – 255 (0 là đen hoàn toàn và 255 là trắng hoàn toàn).

- Với ảnh màu RGB :



Hình 2. 4 Phân bố pixel trên ảnh màu RGB

- Ảnh thật có màu nhìn thấy bằng mắt thường.
- Mỗi pixel trong ảnh màu được biểu diễn bằng 3 giá trị, tương ứng với : RGB (đỏ, xanh lục, xanh dương).
- Mỗi giá trị nằm trong khoảng 0 đến 255 (0 là không có ánh sáng màu đó, 255 là sáng nhất ở kênh màu đó).

❖ Ảnh số :

- Ảnh số là một ma trận 2 chiều (ảnh xám) hoặc 3 chiều (ảnh màu) gồm các điểm ảnh (pixel).
- Mỗi pixel chứa thông tin về độ sáng (ảnh xám) hoặc giá trị màu RGB (ảnh màu).
- Kích thước ảnh: Width (rộng) x Height (cao) và số kênh màu (1 cho ảnh xám hoặc 3 cho RGB).

b. Xử lý ảnh :

Xử lý ảnh là quá trình chuyển đổi một hình ảnh sang dạng kỹ thuật số và thực hiện các thao tác nhất định để nhận được một số thông tin hữu ích từ hình ảnh đó. Hệ thống xử lý hình ảnh thường coi tất cả các hình ảnh là tín hiệu 2D khi áp dụng một số phương pháp xử lý tín hiệu đã xác định trước.

❖ Các loại xử lý hình ảnh chính :

- Nhận diện – Phân biệt hoặc phát hiện các đối tượng trong hình ảnh.
- Làm sắc nét và phục hồi – Tạo hình ảnh nâng cao từ hình ảnh gốc.
- Nhận dạng mẫu – Đo các mẫu khác nhau xung quanh các đối tượng trong hình ảnh.
- Truy xuất – Duyệt và tìm kiếm hình ảnh từ một cơ sở dữ liệu lớn gồm các hình ảnh kỹ thuật số tương tự như hình ảnh gốc.

2.3.2. Tổng quan về thị giác máy tính :

Thị giác máy tính là công nghệ cho phép máy móc tự động phân tích, nhận diện và hiểu nội dung hình ảnh tương tự như cách con người nhìn và suy luận. Công nghệ này kết hợp xử lý ảnh, học máy và trí tuệ nhân tạo, thực hiện các bước sau:

- Thu thập hình ảnh từ camera, cảm biến hoặc cơ sở dữ liệu.
- Tiền xử lý để làm sạch, điều chỉnh độ sáng, tương phản, loại nhiễu.
- Trích xuất đặc trưng như cạnh, góc, màu sắc, kết cấu để đơn giản hóa và làm nổi bật thông tin quan trọng.
- Phân loại đối tượng dựa trên các đặc trưng đã học từ dữ liệu huấn luyện.
- Nhận diện đối tượng: xác định rõ vị trí, loại và số lượng đối tượng trong ảnh.

- Theo dõi đối tượng: theo dõi sự chuyển động của đối tượng qua nhiều khung hình trong video.

Thị giác máy tính đóng vai trò quan trọng trong các hệ thống AI hiện đại như nhận diện lỗi sản phẩm, xe tự lái, y tế thông minh và giám sát an ninh.

2.3.3. Phân biệt xử lý ảnh và thị giác máy tính :

Bảng 2 1 Phân biệt xử lý ảnh và thị giác máy tính

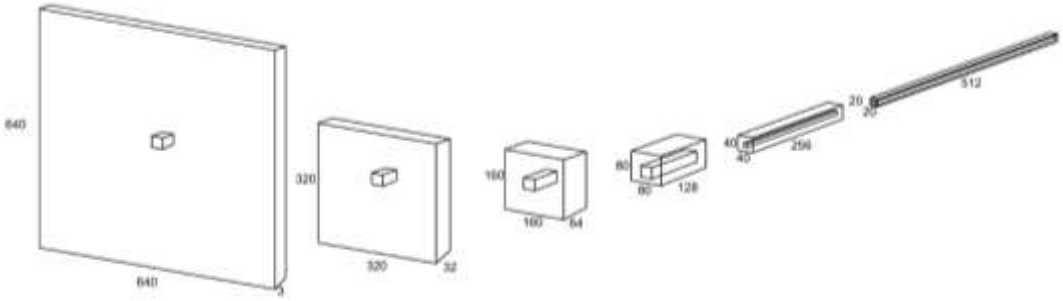
Tiêu chí	Nội dung	
	Xử lý ảnh	Thị giác máy tính
Mục tiêu	Cải thiện chất lượng ảnh, biến đổi ảnh để dễ phân tích hoặc hiển thị	Hiểu và diễn giải nội dung của ảnh hoặc video (giống cách con người nhìn).
Đầu vào	Một ảnh số (digital image)	Ảnh, chuỗi ảnh (video), hoặc khung hình.
Đầu ra	Một ảnh số khác (đã được xử lý	Thông tin, mô tả, quyết định, nhãn phân loại.
Tác vụ chính	- Làm mịn, làm sắc nét, lọc nhiễu - Cắt ảnh, chuyển ảnh sang xám, RGB - Biến đổi ảnh	- Phát hiện vật thể - Nhận diện khuôn mặt, OCR - Dự đoán hành vi, theo dõi đối tượng
Hiểu biết	Không hiểu biết, chỉ xử lý, không hiểu nội dung trong ảnh	Có hiểu biết, hiểu nội dung trong ảnh để đưa ra quyết định
Ví dụ	Làm rõ vùng ảnh bị mờ, cân bằng ánh sáng, lọc nhiễu từ camera	Nhận diện khuôn mặt, xác định lỗi trên sản phẩm công nghiệp, nhận diện biển số xe

2.4. Mô hình mạng thần kinh tích chập CNN :

2.4.1. Tổng quan về CNN :

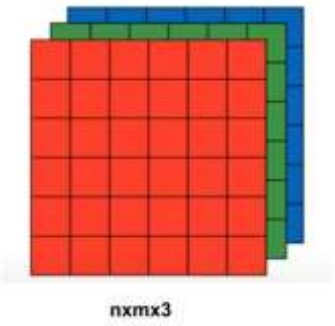
CNN (Convolutional Neural Network) là một mô hình học sâu mạnh mẽ, đặc biệt hiệu quả trong xử lý hình ảnh. CNN hoạt động bằng cách trích xuất đặc điểm từ ảnh qua các lớp tích chập, sau đó đưa dữ liệu qua các lớp kết nối đầy đủ và phân loại bằng hàm Softmax. Kết quả là xác suất đối tượng thuộc các loại khác nhau trong ảnh.

Trong đề tài lần này, ngoài việc nhận diện đối tượng là lỗi gì (trong số những lỗi đã nêu qua ở chương 1) thì còn cần định vị đối tượng (xác định lỗi đó nằm ở vị trí nào trên côn sợi). Nếu sử dụng lớp FC (Fully Connected) trong cấu trục mạng sẽ không tối ưu vì lớp FC (Fully Connected) phục vụ tốt cho việc nhận diện đối tượng nhưng lại kém khi định vị đối tượng. Vì vậy ở đề tài này nhóm em sẽ sử dụng cấu trúc CNN và thay thế các FC (Fully Connected) bằng các Convolution layer để giữ thông tin không gian.



Hình 2. 5 Kích thước ảnh sau khi qua tích chập

2.4.2. Thuật toán CNN :



Hình 2. 6 Lưới ảnh RGB

Ảnh có kích thước ma trận 3 kênh bao gồm :

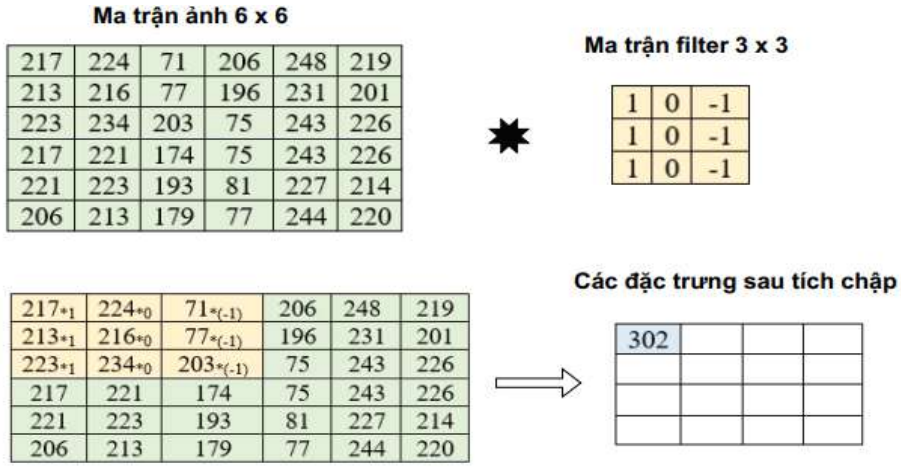
$$R: \begin{bmatrix} a_{r11} & a_{r12} & \dots & a_{r1m} \\ a_{r21} & a_{r22} & \dots & a_{r2m} \\ \dots & \dots & \dots & \dots \\ a_{rn1} & a_{rn2} & \dots & a_{rnm} \end{bmatrix} \quad G: \begin{bmatrix} a_{g11} & a_{g12} & \dots & a_{g1m} \\ a_{g21} & a_{g22} & \dots & a_{g2m} \\ \dots & \dots & \dots & \dots \\ a_{gn1} & a_{gn2} & \dots & a_{gnm} \end{bmatrix}$$

$$B: \begin{bmatrix} a_{b11} & a_{b12} & \dots & a_{b1m} \\ a_{b21} & a_{b22} & \dots & a_{b2m} \\ \dots & \dots & \dots & \dots \\ a_{bn1} & a_{bn2} & \dots & a_{bnm} \end{bmatrix}$$

a. Lớp tích chập Convolution layer :

Convolutional layer là lớp cốt lõi trong CNN, thực hiện các phép tính chính. Các thành phần quan trọng gồm:

- **Filter map**: Ma trận chứa tham số dùng để quét qua ảnh và trích xuất đặc trưng.
- **Stride**: Bước di chuyển của filter trên ảnh.
- **Padding**: Giá trị (thường là 0) thêm vào viền ảnh để giữ nguyên kích thước.
- **Feature map**: Kết quả đầu ra sau khi filter quét ảnh, thể hiện đặc trưng đã được trích xuất.



Hình 2. 7 Phép nhân tích chập

Nhân tích chập ma trận ảnh 3D với lớp filter 3x3x3 để trích xuất đặc trưng, ma trận của lớp filter của 3 lớp :

$$\begin{bmatrix} a'_{11} & a'_{12} & a'_{13} \\ a'_{21} & a'_{22} & a'_{23} \\ a'_{31} & a'_{32} & a'_{33} \end{bmatrix} \quad \begin{bmatrix} a''_{11} & a''_{12} & a''_{13} \\ a''_{21} & a''_{22} & a''_{23} \\ a''_{31} & a''_{32} & a''_{33} \end{bmatrix} \quad \begin{bmatrix} a'''_{11} & a'''_{12} & a'''_{13} \\ a'''_{21} & a'''_{22} & a'''_{23} \\ a'''_{31} & a'''_{32} & a'''_{33} \end{bmatrix}$$

Các thông số trong lớp tích chập :

- Số filter : k = 1
- Kích thước ảnh đầu vào : w x h = n x m
- Kích thước của filter : n x n = 3 x 3
- Đường viền Padding : p = 1
- Bước nhảy Stride : s = 1

Ở đây sử dụng same padding (p=1) để ma trận đầu ra sau filter vẫn bảo toàn kích thước và giữ thông tin biên ảnh. Còn bước nhảy stride s =1 không làm giảm kích thước ảnh ở bước này và dùng MaxPool để giảm kích thước ảnh.

0	0	0	0	0	0
0	a_{r11}	a_{r12}	...	a_{r1m}	0
0	a_{r21}	a_{r22}	...	a_{r1m}	0
0	...	...	...	...	0
0	a_{rn1}	a_{rn2}	...	a_{rnm}	0
0	0	0	0	0	0

Hình 2. 8 Lưới ảnh được sử dụng 1 lớp padding

Tiến hành nhân từng tham số ở từng lớp lại rồi cộng tất cả các tham số lại để đưa ra ma trận mới :

$$a_{11} = 0.a'_{11} + 0.a'_{12} + 0.a'_{13} + 0.a'_{21} + a_{r11}.a'_{22} + a_{r12}.a'_{23} + 0.a'_{31} + a_{r21}.a'_{32} + a_{r22}.a'_{33} + 0.a''_{11} + 0.a''_{12} + 0.a''_{13} + 0.a''_{21} + a_{g11}.a''_{22} + a_{g12}.a''_{23} + 0.a''_{31} + a_{g21}.a''_{32} + a_{g22}.a''_{33} + 0.a'''_{11} + 0.a'''_{12} + 0.a'''_{13} + 0.a'''_{21} + a_{b11}.a'''_{22} + a_{b12}.a'''_{23} + 0.a'''_{31} + a_{b21}.a'''_{32} + a_{b22}.a'''_{33} \quad (2.1)$$

$$a_{12} = 0.a'_{11} + 0.a'_{12} + 0.a'_{13} + a_{r11}.a'_{21} + a_{r12}.a'_{22} + a_{r13}.a'_{23} + a_{r21}.a'_{31} + a_{r22}.a'_{32} + a_{r23}.a'_{33} + 0.a''_{11} + 0.a''_{12} + 0.a''_{13} + a_{g11}.a''_{21} + a_{g12}.a''_{22} + a_{g13}.a''_{23} + a_{g21}.a''_{31} + a_{g22}.a''_{32} + a_{g23}.a''_{33} + 0.a'''_{11} + 0.a'''_{12} + 0.a'''_{13} + a_{g11}.a'''_{21} + a_{b12}.a'''_{22} + a_{b13}.a'''_{23} + a_{b21}.a'''_{31} + a_{b22}.a'''_{32} + a_{b23}.a'''_{33} \quad (2.2)$$

…

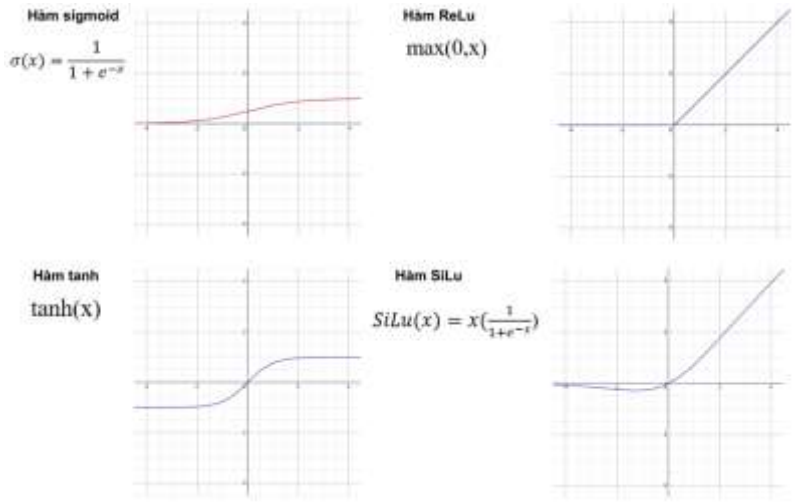
$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

Ma trận mới tạo ra sau khi tích chập với filter có kích thước được tính theo công thức :

$$w_o \text{ x } h_o = \left(\frac{w_i + 2p - n}{s} + 1\right) \text{x} \left(\frac{h_i + 2p - n}{s} + 1\right) \quad (2.3)$$

b. Hàm kích hoạt :

Hàm kích hoạt của một nút định nghĩa đầu ra của nút đó được cho bởi một đầu vào hay tập đầu vào. Hiểu theo cách khác, với một đầu vào hay tập đầu vào, hàm kích hoạt sẽ cho ra đầu ra của một nút bất kỳ nào đó.



Hình 2. 9 Một số hàm kích hoạt

Trong đề tài lần, nhóm em này sử dụng hàm SiLu và hàm Softmax.

Hàm SiLu giúp cải thiện hiệu suất mô hình nhờ tính trơn tru và không đơn điệu, hỗ trợ dòng chảy gradient ổn định và khả năng học các mẫu phi tuyến tính phức tạp.

Hàm Softmax dùng để tính toán xác suất xảy ra của một sự kiện. tính khả năng xuất hiện của một class trong tổng số tất cả các class có thể xuất hiện.

Sau bước tích chập, ma trận kết quả được đưa qua hàm kích hoạt để tăng tính phi tuyến tính. Cuối cùng, bias được cộng vào để điều chỉnh kết quả đầu ra.

Hàm SiLu và hàm softmax :

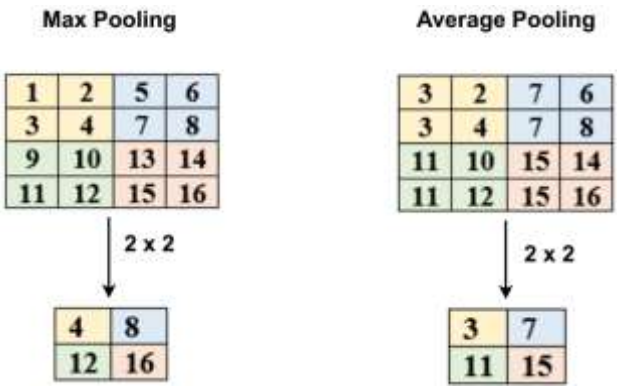
$$\text{SiLu (x)} = \text{x.}\left(\frac{1}{1+e^{-x}}\right) \tag{2.4}$$

$$\text{Softmax(z)} = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \text{với i} = 1, 2, \dots, \text{n} \tag{2.5}$$

c. Lớp gộp Pooling layer :

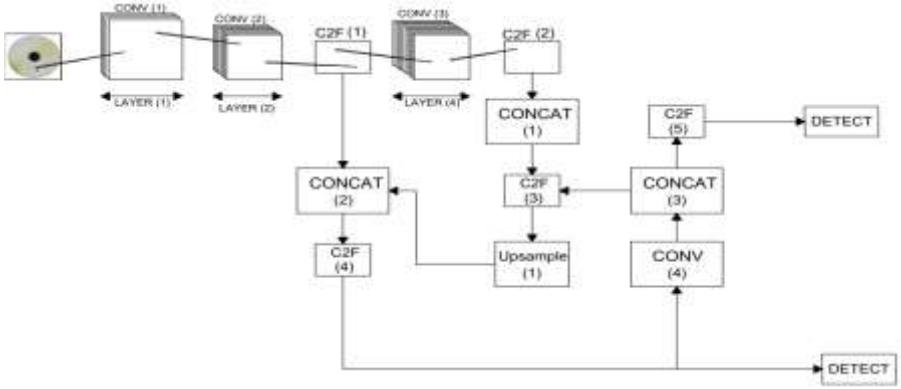
Pooling layer giúp giảm kích thước dữ liệu và số tham số trong mô hình CNN. Có hai loại phổ biến là max pooling và average pooling. Max pooling lấy giá trị lớn nhất trong vùng dữ liệu, trong khi average pooling tính giá trị trung bình. Cả hai đều giúp giảm tải cho mô hình và tăng hiệu quả xử lý.

Trong đó, Max Pooling được dùng phổ biến hơn, vì nó giữ lại đặc trưng nổi bật nhất bằng cách chọn giá trị lớn nhất trong mỗi vùng (thường 2x2). Average Pooling ít được sử dụng hiện nay.



Hình 2. 10 Lớp gộp MaxPooling

d. Minh hoạ cấu trúc mạng học sâu được sử dụng trong đề tài :



Hình 2. 11 Minh hoạ cấu trúc mạng học sâu sử dụng trong đề tài

Ảnh đầu vào có kích thước lớn nên cần đưa qua các lớp tích chập (CONV) để giảm kích thước, giảm số tham số tính toán, loại bỏ chi tiết thừa và làm nổi bật đặc trưng quan trọng. Tuy nhiên, quá trình này cũng làm giảm chất lượng ảnh.

2.4.3. Áp dụng vào tính toán cụ thể :

Từ những lý thuyết và các công thức trên, nhóm em thực hiện trích ma trận ảnh từ *hình 2.3* để tính toán.

a. Convolution :

217	224	71	206	248	219
213	216	77	196	231	201
223	234	203	75	243	226
217	221	174	75	241	213
221	223	193	81	227	214
206	213	179	77	244	220

*

1	**0**	**-1**
1	**0**	**-1**
1	**0**	**-1**

Các tham số : s = 1, p = 0, n = 3. Ma trận đầu ra có kích thước 4 x 4 vì :

w_o x h_0 = ($\frac{wi+2p-n}{s}$ + 1) x ($\frac{hi+2p-n}{s}$ + 1) = ($\frac{6+2.0-3}{1}$ + 1) x ($\frac{6+2.0-3}{1}$ + 1) = 4 x 4

a_{11} = 217.1 + 224.0 + 71.(-1) + 213.1 + 216.0 + 77.(-1) + 223.1 + 234.0 + 203.(-1) = 302

a_{12} = 224.1 + 71.0 + 206.(-1) + 216.1 + 77.0 + 196.(-1) + 234.1 + 203.0 + 75.(-1) = 206

…

a_{43} = 174.1 + … + 244.(-1) = -166

a_{44} = 75.1 + … + 220.(-1) = -414

Ma trận sau khi nhân tích chập với lớp filter :

a_{11} = 302	a_{12} = 206	a_{13} = -371	a_{14} = -169
a_{21} = 199	a_{22} = 325	a_{23} = -435	a_{24} = -294
a_{31} = 91	a_{32} = 447	a_{33} = -141	a_{34} = -422
a_{41} = 98	a_{42} = 424	a_{43} = -166	a_{44} = -414

b. Hàm kích hoạt SiLu :

$$\text{SiLu}(x) = x.(\frac{1}{1+e^{-x}})$$

Với bias = 0.1

SiLu (a_{11}) = a_{11}.($\frac{1}{1+e^{-a11}}$) + b = 302.($\frac{1}{1+e^{-302}}$) + 0.1 = 302,1

SiLu (a_{12}) = a_{12}.($\frac{1}{1+e^{-a12}}$) + b = 206.($\frac{1}{1+e^{-206}}$) + 0.1 = 206,1

…

SiLu (a_{43}) ≈ -1,34.10^{-70} + 0,1

SiLu (a_{44}) ≈ 0 + 0,1

Ma trận sau khi qua hàm kích hoạt SiLu :

a_{11} = 302	a_{12} = 206	a_{13} = 0 + 0,1	a_{14} = -6,8.10^{-72} + 0,1
a_{21} = 199	a_{22} = 325	a_{23} = 0 + 0.1	a_{24} = 0 + 0.1
a_{31} = 91	a_{32} = 447	a_{33} = -8,2.10^{-60} + 0,1	a_{34} = 0 + 0,1
a_{41} = 98	a_{42} = 424	a_{43} = -1,34.10^{-70} + 0,1	a_{44} = 0 + 0,1

c. Lớp gộp MaxPooling :

Sử dụng MaxPooling có kích thước 2x2 :

$$\text{MaxPooling}\begin{pmatrix} 302 & 206 & 0 + 0{,}1 & -6{,}8.10^{-72} + 0{,}1 \\ 199 & 325 & 0 + 0.1 & 0 + 0.1 \\ 91 & 447 & -8{,}2.10^{-60} + 0{,}1 & 0 + 0.1 \\ 98 & 424 & -1{,}34.10^{-70} + 0{,}1 & 0 + 0.1 \end{pmatrix}$$

Ma trận sau lớp gộp MaxPooling : $\begin{pmatrix} 325 & 0 + 0.1 \\ 447 & 0 + 0.1 \end{pmatrix}$

2.5. Xây dựng cấu trúc mô hình học của đề tài :

❖ Kiến trúc mô hình có 3 phần chính :

- **Backbone** : là phần học sâu có vai trò chính là trích xuất những đặc trưng của ảnh đầu vào, giảm kích thước ảnh để giảm chi phí tính toán.
- **Neck** : là phần kết hợp các đặc trưng có được từ nhiều lớp khác nhau của phần Backbone.
- **Head :** là phần dự đoán các classes và bounding box của đối tượng, là đầu ra cuối cùng được tạo bởi mô hình.

2.5.1. Ý nghĩa của các khối được sử dụng trong 3 phần chính :

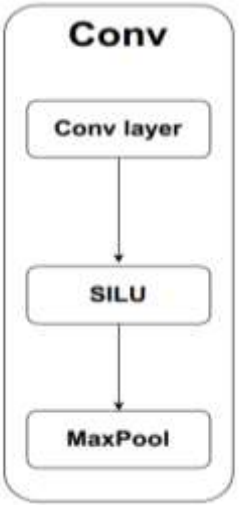
a. Khối tích chập (Conv) :

Trích xuất các đặc trưng quan trọng của ảnh để mô hình để mô hình dễ nhìn và hiểu.

- Conv layer : lớp tích chập, bao gồm các tham số :
  - Bước nhảy stride s
  - Đường viền padding p
  - Kích thước filter kernel k
  - Số lượng kênh filter f
- SiLu : Hàm kích hoạt, công thức :

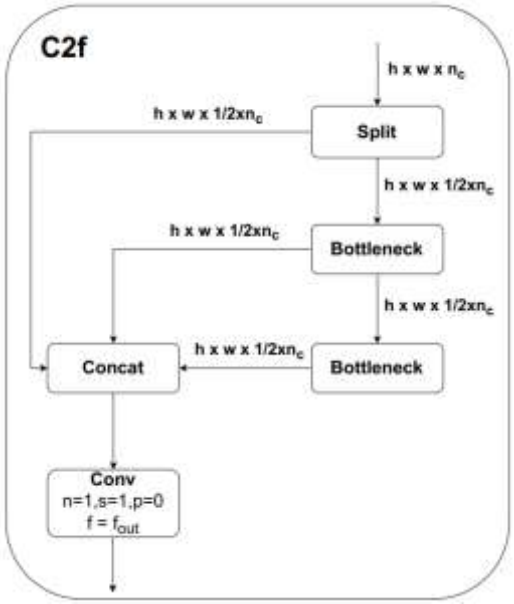
$$\text{SiLu}(x) = x.\left(\frac{1}{1+e^{-x}}\right)$$

- MaxPooling (2 x 2) : Giảm kích thước ảnh đi 2 lần.



Hình 2. 12 Khối tích chập

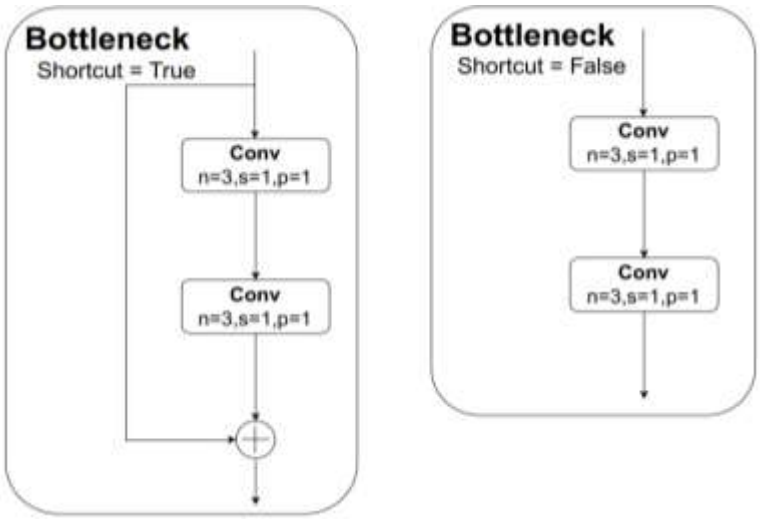
b. Khối C2f :



Hình 2. 13 Khối C2f

Giúp mô hình học tốt các đặc trưng nhờ ghép nhiều thông tin với nhau, trích xuất thêm các thông tin giúp tăng chiều sâu các đặc trưng, có shorcut giúp lan truyền gradient tốt hơn.

- Split : Chia 2 nhánh thành 2 phần bằng nhau có số kênh đi vào mỗi nhánh bằng ½ số kênh input để giảm tính toán khi đi qua khối Bottleneck để mô hình nhẹ hơn, 1 phần sẽ giữ nguyên các đặc trưng ban đầu, 1 phần sẽ trích xuất sâu thêm các đặc trưng
- **Khối Bottleneck** :



Hình 2. 14 Khối Bottleneck

- 2 khối Conv (k=3,s=1,p=1) bên trong khối Bottleneck khi ảnh với các đặc trưng đi qua sẽ giữ nguyên kích thước, học đặc trưng sâu hơn thông qua các tầng Conv. Các tầng Conv trong khối C2f chỉ trích ra các đặc trưng mà vẫn giữ nguyên kích thước ảnh nên không cần sử dụng khối MaxPooling (2x2).
- Shortcut true sẽ giữ lại các đặc trưng gốc, mạng học nhanh hơn khi có thông tin gốc hổ trợ, giúp mô hình lan truyền gradient tốt hơn khi mạng học sâu.
- Shorcut false, mô hình sẽ học các đặc trưng mới mà không có dữ liệu gốc.
- Số lượng khối Bottleneck (m) trong khối C2f được xác đinh theo hệ số độ sâu (d), trong YOLOv8n thì hệ số độ sâu d = 0.33.

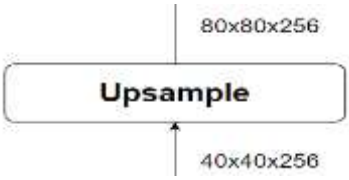
- Với m = 6.d, mô hình sẽ học thêm các đặc trưng nâng cao (giống như các lớp ẩn ở CNN).

Trong đó :

- m : Số bottleneck bên trong C2f để học các đặc trưng sâu hơn
- d : Hệ số độ sâu (Theo mô hình YOLOv8 thì hệ số độ sâu d = 0.33).
- m = 6xd = 6.0.33 = 2 => Có 2 bottleneck.

- Khối Concat sẽ kết nối các thông tin lại với nhau.
- Khối Conv cuối cùng sẽ điều chỉnh số kênh sau khi các thông tin được gộp lại ở concat về số kênh đầu ra như mong muốn.

c. Khối Upsample :



Hình 2. 15 Khối Upsample

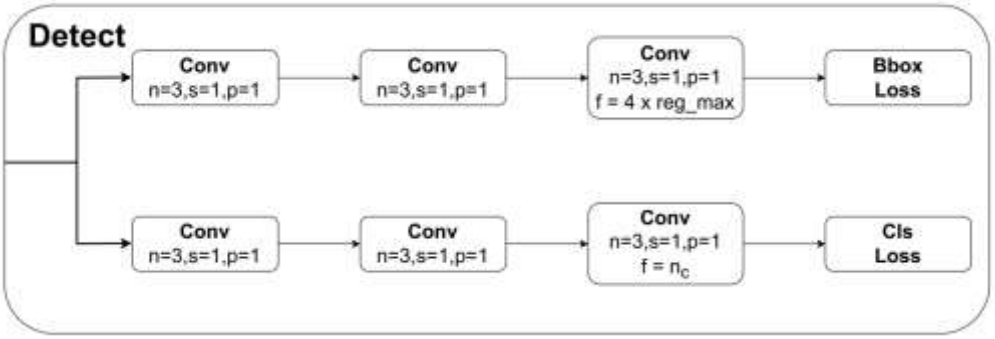
Khối Upsample thực hiện tăng kích thước đặc trưng đầu vào, bằng cách gấp đôi chiều cao và chiều rộng, khôi phục thông tin và độ phân giải cao để mô hình xử lý chi tiết hơn.

d. Khối Concat :

Hình 2. 16 Khối Concat

Khối Concat kết hợp các đặc trưng từ nhiều tầng theo chiều kênh, tăng cường thông tin, giữ lại các đặc trưng cũ và trộn thông tin từ các tầng có độ phân giải khác nhau.

e. Khối Detect :



Hình 2. 17 Khối Detect

Các tầng Conv trong khối Detect chỉ để tinh chỉnh các đặc trưng nhưng không làm thay đổi kích thước ảnh nên không cần sử dụng khối MaxPooling (2x2).

- Nhánh 1 : Dự đoán Bounding Box.
  - Các Conv (k=3,s=1,p=1) sẽ tiếp tục trích xuất để tinh chỉnh các đặc trưng không gian chính xác hơn để dự đoán toạ độ.
  - Khối Conv (k=1,s=1,p=0,f=4.reg_max) dự đoán bounding box (x,y,w,h).
    - 4 là 4 toạ độ left,top,right,bottom.

      Trong đó : Left là toạ độ cạnh trái bounding box ứng xuống trục hoành.

      Top là toạ độ cạnh trên bounding box ứng qua trục tung.

      Right là toạ độ cạnh phải bounding box ứng xuống trục hoành.

      Bottom là toạ độ cạnh dưới bounding box ứng qua trục tung.

    - reg_max là giá trị rời rạc để mô hình hoá toạ độ box (YOLOv8n, reg_max = 4).

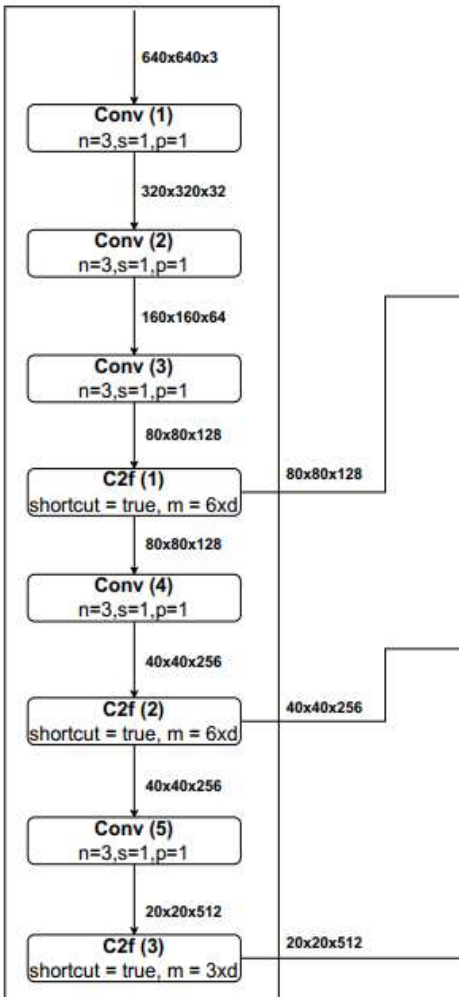
Dfl_loss đưa ra từ mô hình cho thấy được dự đoán toạ độ của Bouding box, dfl_loss càng thấp thì mô hình dự đoán càng chính xác.

- Nhánh 2 : Dự đoán Classes.
  - Các Conv (k=3,s=1,p=1) sẽ tiếp tục trích xuất để tinh chỉnh và làm giàu các đặc trưng dùng cho phân lớp.
  - Khối Conv (k=1,s=1,p=0,f = n_c) giảm số kênh về đúng với số lớp cần dự đoán (n_c).

Cls_loss đưa ra từ mô hình cho thấy được dự đoán phân loại đối tượng, thuộc lỗi nào, cls_loss càng thấp thì mô hình dự đoán càng chính xác.

2.5.2. Cấu trúc từng phần của đề tài :

a. Phần Backbone :



Các khối được sử dụng trong phần Backbone :

- Khối tích chập (Conv).
- Khối C2f.

Phần Backbone là phần tích chập quan trọng nhất trong cấu trúc.

Input đầu vào là một tấm hình – một lưới ma trận đại diện cho từng giá trị sáng tối của từng pixel một. Khi ảnh qua layer đầu tiên, hình sẽ được nén lại, mô hình sẽ nhận diện ra những đặc điểm đơn giản trước, dần dần tổng hợp những đặc điểm đó để tạo ra những đường nét phức tạp hơn. Sau khi đi qua một vài lớp như vậy, tấm hình ban đầu chuyển thành một tập các đặc trưng ở nhiều độ phân mà mô hình có thể học, nhận diện và đánh dấu. Nhờ vậy, mỗi lần người dùng gửi ảnh lên sẽ cho ra kết quả mong muốn.

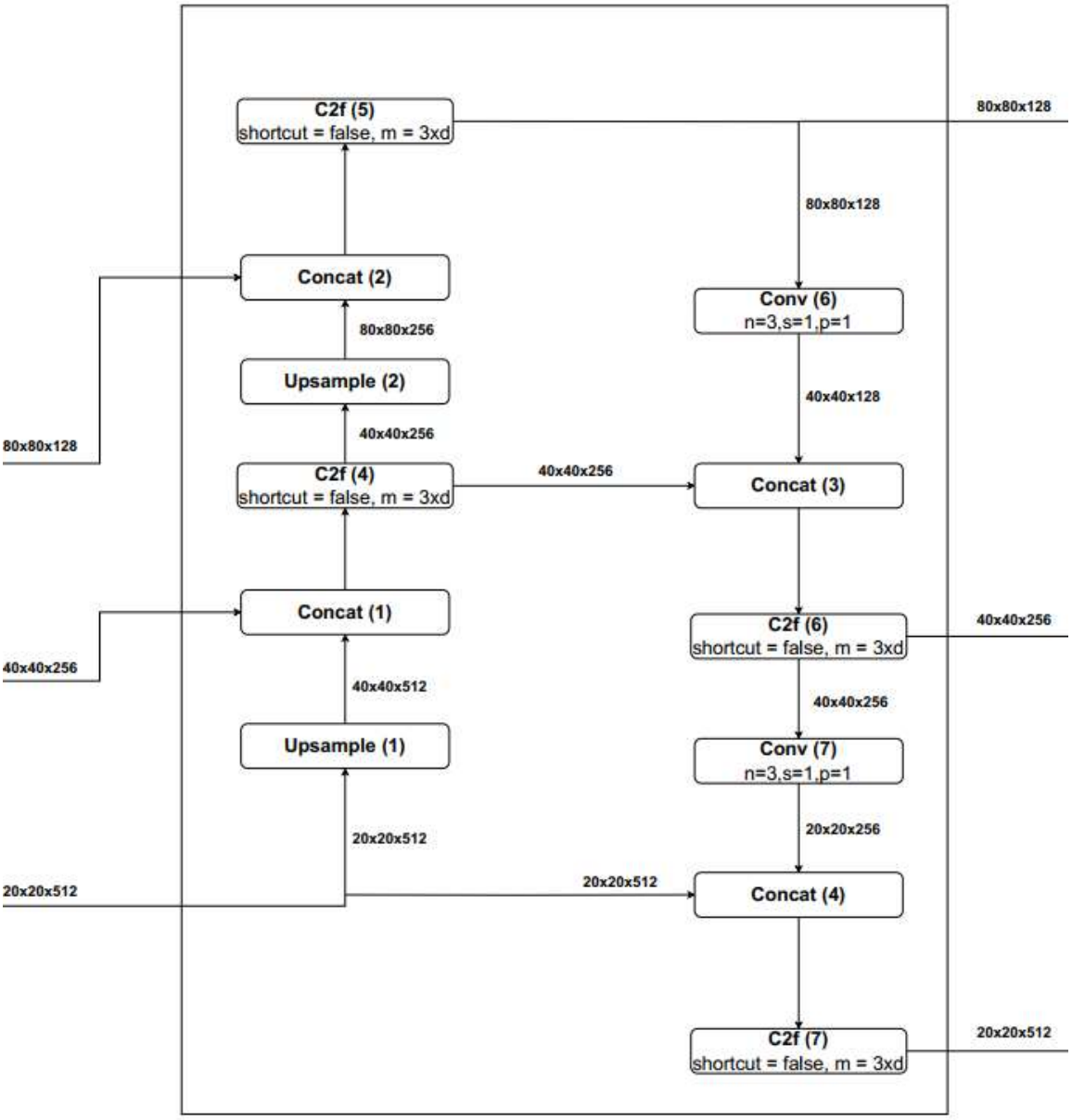
Hình 2. 18 Phần Backbone

Với ảnh đầu vào kích thước là 640 x 640 x 3, xây dựng 3 khối tích chập (Conv (1), (2), (3)) để đưa ảnh về kích thước 80 x 80 x 128. Khối C2f (1) đặt sau khối Conv (3) để giúp mô hình học sâu các đặc trưng cơ bản từ các khối trên và được tổng hợp qua phần Neck đưa ra khối Detect để nhận các lỗi nhỏ.

Từ khối C2f (1) tiếp tục đưa vào khối Conv (4) để trích xuất các đặc trưng phức tạp hơn và thu nhỏ kích thước ảnh còn 40 x 40 x 256. Khối C2f (2) đặt sau khối Conv (4) để giúp mô hình học sâu các đặc trưng phức tạp và được tổng hợp qua phần Neck đưa ra khối Detect để nhận các lỗi trung bình.

Cuối cùng, ảnh với kích thước 40 x 40 x 256 qua khối Conv (5) trích xuất các đặc trưng phức tạp theo chiều sâu, kích thước ảnh giảm xuống 20 x 20 x 512. Khối C2f (3) đặt sau khối Conv (5) để giúp mô hình học các đặc trưng theo chiều sâu và được tổng hợp qua phần Neck đưa ra khối Detect để nhận các lỗi lớn.

b. Phần Neck :



Hình 2. 19 Phần Neck

Các khối được sử dụng trong phần Neck:

- Khối tích chập (Conv).
- Khối C2f.

- Khối Concat.
- Khối Upsample.

Phần Neck là phần tổng hợp các đặc trưng theo nhiều độ phân giải khác nhau từ phần Backbone.

Ở phần Neck, ảnh từ khối C2f (3) với kích thước 20 x 20 x 512 được đưa vào khối Upsample (1) để phóng to kích thước không gian lên 2 lần của đặc trưng ảnh đầu ra, khôi phục thông tin và độ phân giải cao để mô hình xử lý chi tiết hơn.

Khối Concat (1) tổng hợp thông tin nhận được từ khối Upsample (1) và đặc trưng của khối C2f (2) đưa vào khối C2f (4). Sau khi ảnh được kết hợp tại khối Concat (1) thì rất giàu thông tin nhưng chưa được học trộn lẫn nhau nên sẽ được đưa qua khối C2f (4) để tiến hành học sâu các thông tin này.

Khối Upsample (2) nhận thông tin từ khối C2f (4) để phóng to kích thước không gian lên 2 lần của đặc trưng ảnh đầu ra, khôi phục thông tin và độ phân giải cao để mô hình xử lý chi tiết hơn.

Khối Concat (2) tổng hợp thông tin nhận được từ khối Upsample (2) và đặc trưng của khối C2f (1) đưa vào khối C2f (5). Sau khi ảnh được kết hợp tại khối Concat (2) thì rất giàu thông tin nhưng chưa được học trộn lẫn nhau nên sẽ được đưa qua khối C2f (5) để tiến hành học sâu các thông tin này. Đầu ra khối C2f (5) sẽ đưa vào khối Conv (6) và được đưa vào khối Detect ở phần Head.

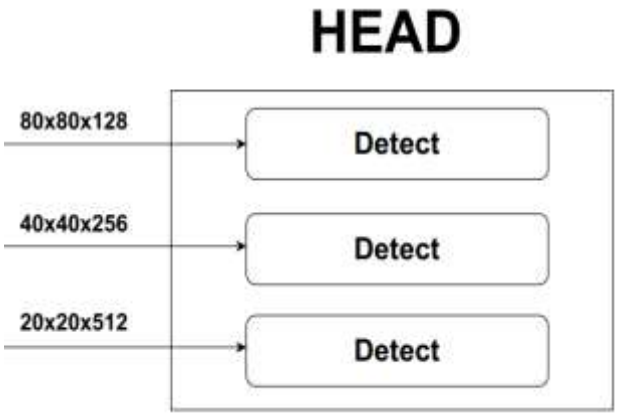
Khối Conv (6) đưa ảnh về kích thước 40 x 40 x 128.

Khối Concat (3) tổng hợp thông tin nhận được từ khối Conv (6) và đặc trưng của khối C2f (4) đưa vào khối C2f (6). Sau khi ảnh được kết hợp tại khối Concat (3) thì rất giàu thông tin nhưng chưa được học trộn lẫn nhau nên sẽ được đưa qua khối C2f (6) để tiến hành học sâu các thông tin này. Đầu ra khối C2f (6) sẽ đưa vào khối Conv (7) và được đưa vào khối Detect ở phần Head.

Khối Conv (7) đưa ảnh về kích thước 20 x 20 x 256.

Khối Concat (4) tổng hợp thông tin nhận được từ khối Conv (7) và đặc trưng của khối C2f (3) đưa vào khối C2f (7). Sau khi ảnh được kết hợp tại khối Concat (4) thì rất giàu thông tin nhưng chưa được học trộn lẫn nhau nên sẽ được đưa qua khối C2f () để tiến hành học sâu các thông tin này. Đầu ra khối C2f (7) được đưa vào khối Detect ở phần Head.

c. Phần Head :



Hình 2. 20 Phần Head

Các khối được sử dụng trong phần Head :

- Khối detect

Phần Head là nơi thực hiện các dự đoán cuối cùng : gồm dự đoán toạ độ bounding box, đối tượng và mức độ tin cậy.

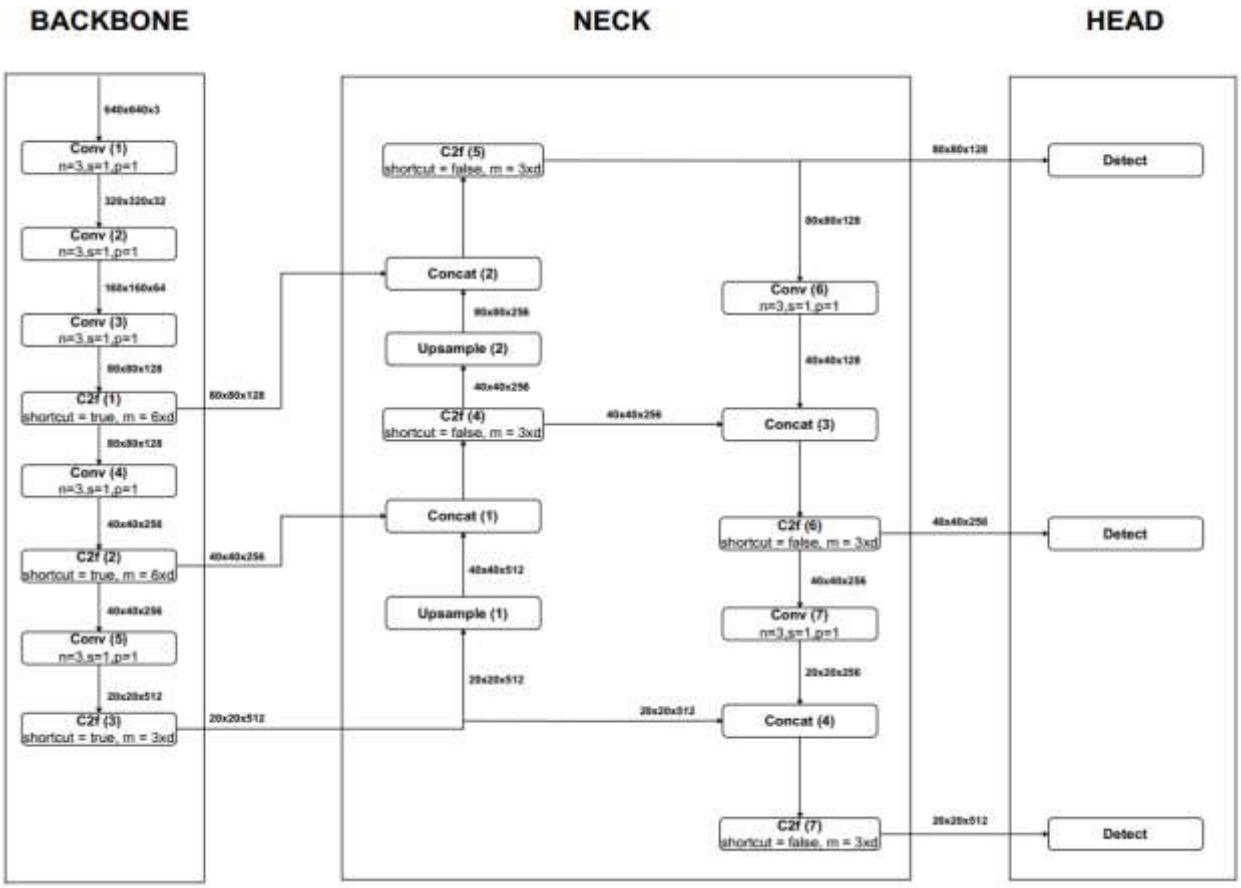
Mỗi khối Detect sẽ nhận diện vật thể ở nhiều kích thước với khác tỷ lệ không gian khác nhau.

Khối Detect (đầu vào 80 x 80 x 128) sẽ phát hiện các vật thể nhỏ nên cần độ phân giải cao.

Khối Detect (đầu vào 40 x 40 x 256) sẽ phát hiện các vật thể trung bình vì không cần độ phân giải quá cao cùng như qua nhiều thông tin đặc trưng.

Khối Detect (đầu vào 20 x 20 x 512) sẽ phát hiện các vật thể lớn nên cần thông tin đặc trưng nhiều, không cần độ phân giải cao.

2.5.3. Cấu trúc mô hình huấn luyện của đề tài :



Hình 2. 21 Cấu trúc mô hình huấn luyện

Sau khi xây dựng từng phần, nhóm tiến hành kết hợp các phần trên và đưa ra cấu trúc hoàn chỉnh để mô hình học và nhận diện lỗi.

2.6. Mô hình nhận dạng vật thể (thuật toán) YOLOv8 :

YOLOv8 là một thuật toán thị giác máy tính được sử dụng để phát hiện đối tượng, được thiết kế để phát hiện đối tượng trong thời gian thực và có thể xử lý hình ảnh trong vài mili giây.

❖ Lý do lựa chọn YOLOv8 :

- Độ chính xác cao.
- Tốc độ nhanh.
- Dễ sử dụng.
- Phù hợp phát hiện đối tượng trong đề tài.

2.6.1. Tổng quan về YOLOv8 :

a. Khái quát về hệ thống phát hiện đối tượng :

Phát hiện đối tượng là một công nghệ máy tính liên quan đến thị giác máy tính và xử lý hình ảnh nhằm phát hiện các trường hợp của đối tượng ngữ nghĩa của một lớp nhất định trong hình ảnh và video kỹ thuật số.

Phát hiện đối tượng được coi là một trong những lĩnh vực quan trọng nhất trong phát triển của học sâu (Deep Learning) và xử lý hình ảnh.

b. Giới thiệu về YOLOv8 :

Yolo là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers (Conv) và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers (FC) sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh.

2.6.2. Các bước để huấn luyện YOLOv8 :

a. Chuẩn bị dữ liệu huấn luyện :

Để huấn luyện YOLOv8, cần tập dữ liệu gồm hình ảnh và chú thích (class và bounding box). Việc tạo dữ liệu tùy chỉnh thường tốn nhiều thời gian, nhưng hiện nay đã có các công cụ hỗ trợ như LabelImg, RectLabel và Roboflow giúp gắn nhãn và xử lý dữ liệu dễ dàng hơn.

b. Các Bước Tạo Dataset :

Thu thập dữ liệu : Đầu tiên, cần xác định vấn đề và loại dữ liệu cần thu thập, có thể qua chụp ảnh, tải từ internet hoặc nguồn khác.

Gắn nhãn cho dữ liệu : Sau khi ta đã thu thập dữ liệu, tiến hành gắn nhãn dữ liệu bằng cách chú thích vị trí và loại đối tượng trong ảnh, sử dụng các công cụ hỗ trợ.

c. Quá trình huấn luyện :

Việc cài đặt tham số rất quan trọng vì ảnh hưởng trực tiếp đến hiệu suất và độ chính xác của mô hình. Các tham số chính gồm:

- Batch size: Số mẫu được đưa vào mô hình mỗi lần lặp, ảnh hưởng đến tốc độ và hiệu quả huấn luyện.
- Learning rate: Tốc độ cập nhật trọng số; quá lớn dễ gây mất ổn định, quá nhỏ có thể làm chậm quá trình học.

- Epochs: Số lần toàn bộ dữ liệu được huấn luyện qua mô hình, giúp mô hình cải thiện dần độ chính xác.

2.7. Chuẩn bị dữ liệu cho mô hình tự học :

2.7.1. Các bước chuẩn bị dữ liệu và tham số cho mô hình :

Hệ thống triển khai cơ chế tự học theo thời gian thực, quá trình học theo các bước như sau:

B1 : Upload tệp zip có chứa :

- Ảnh lỗi
- Nhãn lỗi định dạng YoLo
- Tên lỗi trong file Classes.txt

B2 : Hệ thống kiểm tra dữ liệu, tạo file **data.yaml**, và hợp nhất các dataset cũ.

B3 : Gọi mô hình YOLOv8 được huấn luyện lại trên toàn bộ tập hợp dữ liệu (**merged_dataset**) với các tham số **epochs = 50** , **ingsz = 640.**

B4 : Sau khi huấn luyện, hệ thống sẽ tạo ra mô hình (**best.pt**).

❖ **Trong đề tài sẽ chuẩn bị dữ liệu để mô hình tự học như sau :**

- Chuẩn bị 4 file zip chứa 4 dataset để huấn luyện.
- Giai đoạn đầu tiên sẽ đưa lỗi *Tách xơ ngoại lai*, có lẫn một sợi chỉ nhỏ để học.

Hình 2. 22 Lỗi tách xơ ngoại lai nhỏ

- Khi học xong sẽ tiến hành test thử mô hình tạo ra bằng cách đưa dataset cuộn sợi có lẫn sợi chỉ màu nhỏ và cuộn sợi có lẫn sợi chỉ to và dài hơn. Mô hình test được cho thấy nhận diện tốt những lỗi sợi chỉ màu nhỏ còn những sợi chỉ to và dài hơn thì không nhận diện được.
- Tiếp theo sẽ đưa vào dataset cuộn sợi có lẫn chỉ màu to và dài hơn để học.

Hình 2. 23 Lỗi tách xơ ngoại lai lớn

- Sau đó sẽ lấy mô hình học xong để tiếp tục test kết quả. Kết quả test được cho thấy khi bỏ những sợi chỉ màu lẫn ngắn hay dài bao nhiêu trên cuộn sợi thì mô hình vẫn nhận diện tốt.
- Giai đoạn tiếp theo sẽ đưa lỗi *Sai thành phần* vào để mô hình học, trước đó cũng sẽ test thử mô hình có nhận diện được lỗi này không vì chưa được học thì mô hình chưa nhận diện được.
- Cũng tương tự như lỗi *Tách xơ ngoại lai* thì sẽ đưa lỗi lẫn một chút thành phần khác của côn sợi vào để mô hình học và test thì nhận diện tốt nhưng khi đưa lỗi lẫn thành phần nhiều hơn thì mô hình vẫn chưa nhận diện được.

Hình 2. 24 Lỗi sai thành phần nhỏ

- Tiếp tục đưa lỗi lẫn nhiều thành phần hơn vào để học

Hình 2. 25 Lỗi sai thành phần lớn

- Mô hình đưa cuối cùng sau khi đưa tệp zip các lỗi lên để học được test thì thấy mô hình tự học tăng dần rất tốt, nhận diện được cả lỗi *Sai thành phần* và lỗi *Tách xơ ngoại lai* riêng biệt khi đưa cả hai lỗi vào nhận diện.

2.7.2. Cơ chế học tăng dần :

Bảng 2 2 Cơ chế học tăng dần của mô hình

Đặc điểm	Mô tả
Mục tiêu	Mô hình học kiến thức mới mà không mất kiến thức cũ
Cách triển khai	Mỗi lần upload dataset mới, hệ thống gộp toàn bộ dữ liệu cũ và mới để huấn luyện lại
Lợi ích	Linh hoạt cập nhật lỗi mới Phù hợp với thực tế
Hạn chế	Nếu dữ liệu lỗi cũ quá ít, mô hình có thể sẽ quên lỗi

2.7.3. Hiệu chỉnh mô hình :

- Sử dụng mô hình YOLOv8n (nhẹ nhanh).
- Nếu accuracy thấp => dùng YOLOv8s hoặc YOLOv8m.
- Tăng epochs từ 50 lên 100 hoặc hơn nếu có nhiều dữ liệu.

2.8. Tổng quan về điện toán đám mây (Cloud Computing) :

2.8.1. Khái niệm :

Điện toán đám mây (Cloud Computing) là mô hình điện toán cho phép người dùng truy cập và sử dụng tài nguyên máy tính (máy chủ, lưu trữ, phần mềm…) thông qua Internet. Thay vì phải đầu tư phần cứng riêng, doanh nghiệp có thể sử dụng dịch vụ từ nhà cung cấp cloud như AWS, Azure, Google Cloud.

2.8.2. Các mô hình triển khai Cloud :

a. Public Cloud :

Hạ tầng cloud được chia sẻ công khai cho nhiều tổ chức hoặc người dùng qua Internet.

Do bên thứ ba (như Amazon, Microsoft, Google) quản lý và cung cấp.

Người dùng chỉ cần đăng ký tài khoản, trả phí theo mức sử dụng (pay-as-you-go), không cần đầu tư phần cứng.

b. Private Cloud :

Hạ tầng được xây dựng riêng biệt cho một tổ chức, không chia sẻ với bên ngoài.

Do doanh nghiệp tự quản lý hoặc thuê đơn vị chuyên trách triển khai và duy trì.

c. Hybrid Cloud :

Kết hợp giữa public cloud và private cloud.

Dữ liệu nhạy cảm, quan trọng lưu trữ trên private cloud; các tác vụ linh hoạt như xử lý ảnh, training AI thực hiện trên public cloud.

2.8.3. Ứng dụng Cloud vào hệ thống của đề tài :

Bảng 2 3 Các dịch vụ Cloud được sử dụng trong đề tài

Dịch vụ AWS	Vai trò trong đề tài	Chức năng chi tiết
EC2 (Elastic Compute Cloud)	Chạy flask và xử lý nhận diện ảnh	- Máy chủ ảo được sử dụng để triển khai backend - EC2 nhận ảnh từ người dùng, giải nén, gọi mô hình YOLO để nhận diện lỗi
S3 (Simple Storage Service)	Lưu trữ ảnh gốc và kết quả ảnh	- Ảnh gốc do người dùng tải lên được lưu trong **uploads**, ảnh đã sử lý lưu vào **results** - Có thể lưu trữ hàng ngàn ảnh với độ bền bảo mật cao
DynamoDB	Quản lý lịch sử nhận diện ảnh	- Lưu trữ thông tin mỗi lần nhận diện : Tên ảnh, loại lỗi, thời gian xử lý, người gửi. - Giúp phân tích và truy xuất lỗi theo thời gian thực
IAM (Identity and Access Management)	Quản lý quyền truy cập và các dịch vụ	- Tạo tài khoản truy cập riêng biệt cho từng thành phần hệ thống - Gán quyền cụ thể từ S3, DynamoDB cho EC2 để ngăn chặn quyền truy cập trái phép

2.9. Các nền tảng, phần mềm được sử dụng cho hệ thống :

Với cấu trúc và nguyên lý hoạt động đã trình bày ở các mục trên, nhóm sẽ đề xuất các nền tảng, phần mềm và một số thư viện như sau để tiến thành thực hiện đề tài lần này :

2.9.1. Ngôn ngữ lập trình Python :

❖ **Ứng dụng cho đề tài :**

Trong đề tài lần này, ngôn ngữ lập trình chính được sử dụng là python, dùng để xây dựng và huấn luyện mô hình AI (YOLOv8), xây dựng websever và kết nối triển khai trên hệ thống cloud (AWS).

2.9.2. Ngôn ngữ lập trình HTML :

❖ **Ứng dụng cho đề tài :**

- HTML được dùng để tạo giao diện frontend cho người dùng trên websever.
- Cung cấp giao diện để các công ty gửi data, người dùng gửi ảnh, nhận kết quả và tải kết quả về.

2.9.3. Phần mềm Pycharm :

❖ **Ứng dụng cho đề tài :**

Phần mềm pycharm làm môi trường để lập trình ngôn ngữ python, tích hợp quản lý môi trường ảo (virtualenv), cài đặt thư viện, tự động hoàn thành mã lệnh, phát triển web với flask và tích hợp với cloud và để test ảnh nhận diện và flask api trước khi đưa lên cloud.

2.9.4. Google Colab :

❖ **Ứng dụng cho đề tài :**

Để huấn luyện mô hình YOLOv8 thì phải cần cấu hình máy có GPU khá cao nên google colab có vai trò quan trọng trong việc huấn luyện và tiết kiệm thời gian nhờ có GPU ảo mạnh.

2.9.5. Google Drive :

❖ **Ứng dụng cho đề tài :**

Google Drive được dùng đến vì Google Colab được sử dụng miễn phí nên cứ mỗi 24 tiếng sẽ reset dữ liệu và bị mất hết dữ liệu đã huấn luyện, vậy nên sử dụng Google Drive là để liên kết với Google Colab để dữ liệu được uploads lên từ các công ty và mô hình được tạo ra mỗi lần huấn luyện sẽ lưu trữ vào trong Google Drive nên cứ mỗi lần Google Colab reset thì dữ liệu sẽ không bị mất.

2.9.6. Ngrok :

❖ **Ứng dụng cho đề tài :**

Khi flask API để tạo ra đường link cho các công ty gửi data, đường link này sẽ là localhost. Ngrok được đưa vào để chuyển đổi đường link localhost này sang link Public để có thể truy cập được ở bất cứ đâu.

2.9.7. Cloud AWS :

❖ **Ứng dụng cho đề tài :**

- AWS(Amazon Web Sever) có vai trò làm nền tảng cloud chính để triển khai hệ thống phát hiện lỗi sợi, lưu trữ dữ liệu và phục vụ cho người dùng từ xa.
- Có bảo mật và truy cập an toàn thông qua IAM và khoá truy cập.
- Cho phép người dùng, doanh nghiệp hoặc các đối tác gửi ảnh lỗi sợi lên và nhận kết quả nhanh chóng.

Các dịch vụ được sử dụng trong AWS :

- EC2 : Máy chủ chạy backend flask 24/7.
- S3 : Lưu trữ ảnh gốc và ảnh kết quả sau xử lý.
- DynamoDB : Lưu lịch sử xử lý và thông tin người dùng .
- IAM : Để cấp quyền từ S3 và DynamoDB cho EC2.

2.9.8. Phần mềm Termius :

❖ **Ứng dụng cho đề tài :**

- Trong đề tài này, Termius đóng vai trò quan trọng trong việc điều khiển và giám sát máy chủ triển khai hệ thống nhận diện lỗi sợi.
- Termius kết nối SSH tới AWS EC2: Đăng nhập vào máy chủ EC2 một cách dễ dàng, an toàn.
- Chạy và kiểm tra Flask API: Sử dụng Termius để điều khiển server, chạy ứng dụng Flask và kiểm tra hoạt động của API.

2.10. Kết luận chương :

Ở chương 2, nhóm em đã xây dựng xong mô hình để chẩn đoán kỹ thuật cho các nhà máy công nghiệp sợi. Nhóm đã đưa ra cấu trúc, nguyên lý hoạt động và các phần mềm cần thiết để tiến hành xây dựng mô hình, nêu ra được các lợi ích khi sử dụng phần mềm đó để áp dụng cho đề tài. Tuy nhiên việc sử dụng các phần mềm này vẫn còn nhiều hạn chế. Nhóm đã lên các phương án khắc phục để hệ thống được hoạt động tối ưu nhất.

Chương 3 nhóm sẽ đi sâu vào thiết kế cụ thể, từng bước làm trọng tâm cho đối tượng chính của đề tài lần này.

CHƯƠNG 3 : THIẾT KẾ CỤ THỂ CHO MÔ HÌNH ĐÁNH GIÁ CHỈ TIÊU CHẤT LƯỢNG SẢN XUẤT SỢI

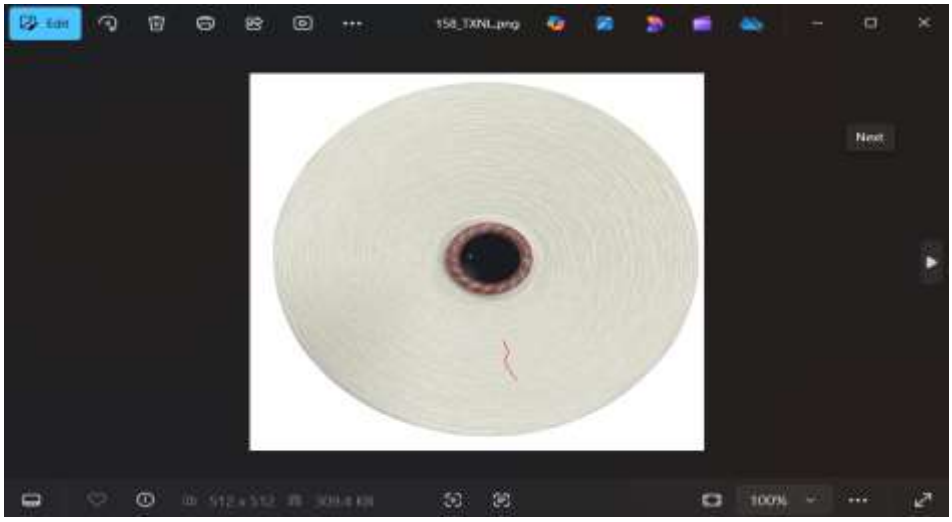
3.1. Nhiệm vụ trọng tâm :

Như giải pháp đã đưa ra ở chương 1 và mô hình xây dựng ở chương 2 thì ở chương này, nhóm sẽ tập trung đi sâu vào trọng tâm cách mô hình tự học, xử lý ảnh, từng bước làm và thiết kế cụ thể đối tượng :

- Phân tích cách học và các phương trình toán học trọng tâm của mô hình :
- Xây dựng và thiết kế phần mềm của mô hình :
  - Xây dựng websever để người dùng truy cập.
  - Thiết kế giao diện frontend để gửi dataset và ảnh nhận diện và tải ảnh nhận diện về.
  - Lập trình cho mô hình có khả năng học tăng dần.
  - Xây dựng cloud và tạo các dịch vụ cần thiết để nhận diện và lưu lịch sử.
  - Tối ưu chi phí các dịch vụ được sử dụng trên cloud.

3.2. Phân tích cách học của mô hình nhận diện YOLO :

3.2.1. Phân tích cách học :



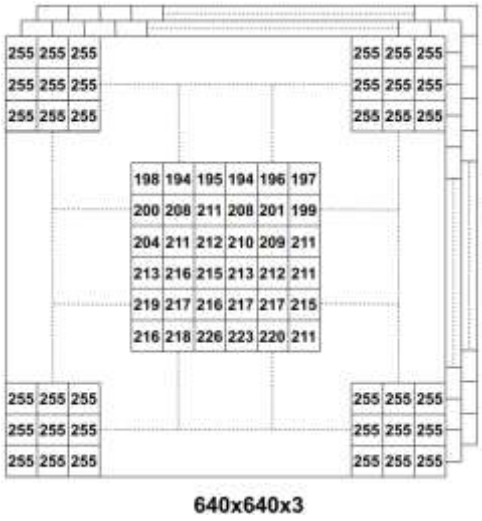
Hình 3. 1 Ảnh cuộn sợi lỗi thực tế

Ảnh đưa vào được resize theo kích thước 640x640, ảnh RGB nên có 3 lớp kênh màu nên ảnh sẽ có kích thước 3D 640x640x3 và đưa vào mô hình để học.

Mỗi bức ảnh là một ma trận mà trong đó các điểm ảnh là tham số của ma trận.

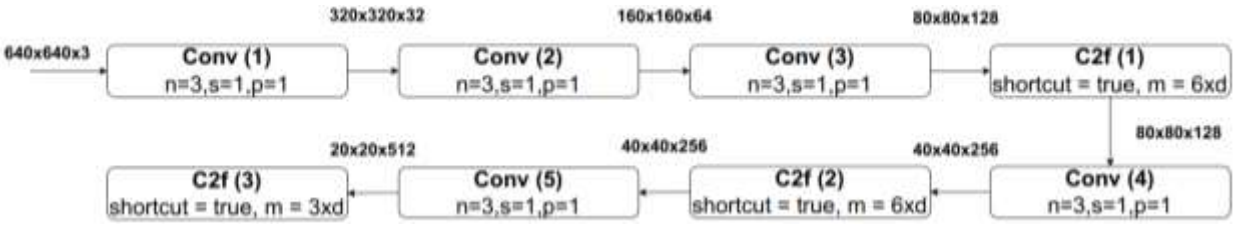
Ở *hình 3.1* , phông nền bao quanh côn sợi có màu trắng, theo không gian màu RGB thì các điểm ảnh tại vùng này ở cả 3 kênh là R: 255,G: 255,B: 255. Các điểm ảnh trên côn sợi ở 3 kênh dao động trong khoảng từ [200;253]. Giá trị điểm ảnh ở cả 3 kênh tại lõi của côn sợi dao động trong khoảng từ 15 đến 170. Lỗi trên côn sợi là 1 sợi chỉ đỏ, giá trị điểm ảnh tại điểm này là R : [220;255], G,B : [10;200].

Dưới đây trích một vài điểm ảnh ở kênh màu đỏ (R) được lấy từ hình 3.1



Hình 3. 2 Lưới ảnh từ hình 3.1

3.2.1.1. Phần Backbone :



Hình 3. 3 Phần Backbone

a. Lớp Conv (1):

Các thông số ở lớp tích chập đầu tiên :

- Số filter : k = 32
- Kích thước ảnh đầu vào : w x h x n_c= 640 x 640 x 3
- Kích thước của filter : n x n x n_c = 3 x 3 x 3
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập đầu tiên :

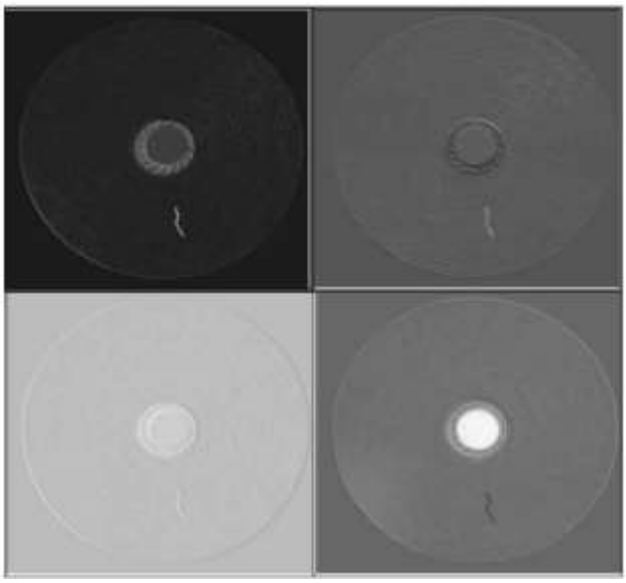
$$\left[\frac{w+2p-n}{s}+1\right] \text{x} \left[\frac{h+2p-n}{s}+1\right] \text{x k} = \left[\frac{640+2.1-3}{1}+1\right] \text{x} \left[\frac{640+2.1-3}{1}+1\right] \text{x 32}$$

$$w_1\text{'} \text{ x } h_1\text{'} \text{ x k} = 640 \text{ x } 640 \text{ x } 32 \quad (3.1)$$

Sau đó ma trận ảnh sẽ được đưa vào hàm kích hoạt SiLu.

Sau khi qua lớp MaxPooling, kích thước ảnh đầu ra ở lớp 1 sẽ là :

$$w_1 \text{ x } h_1 \text{ x k} = 320 \text{ x } 320 \text{ x } 32$$



Hình 3. 4 Một số đặc trưng trong 32 lớp

b. Lớp Conv (2) :

Các thông số ở lớp tích chập thứ 2 :

- Số filter : k = 64
- Kích thước ảnh đầu vào : $w_1 \text{ x } h_1 \text{ x } n_{c1} = 320 \text{ x } 320 \text{ x } 32$
- Kích thước của filter : $n \text{ x } n \text{ x } n_{c1} = 3 \text{ x } 3 \text{ x } 32$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 2 :

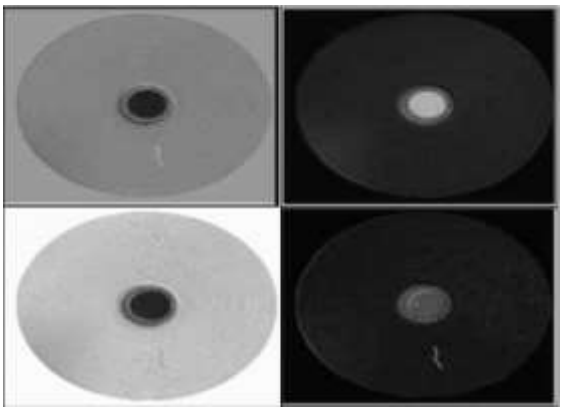
$$\left[\frac{w1+2p-n}{s}+1\right] \text{x} \left[\frac{h1+2p-n}{s}+1\right] \text{x k} = \left[\frac{320+2.1-3}{1}+1\right] \text{x} \left[\frac{320+2.1-3}{1}+1\right] \text{x 64}$$

$$w_2\text{'} \text{ x } h_2\text{'} \text{ x k} = 320 \text{ x } 320 \text{ x } 64 \quad (3.2)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Sau khi qua lớp MaxPooling, kích thước ảnh đầu ra ở lớp 2 sẽ là :

$$w_2 \text{ x } h_2 \text{ x k} = 160 \text{ x } 160 \text{ x } 64$$



Hình 3. 5 Một số đặc trưng trong 64 lớp

c. Lớp Conv (3) :

Các thông số ở lớp tích chập thứ 3 :

- Số filter : k = 128
- Kích thước ảnh đầu vào : w_2 x h_2 x n_{c2} = 160 x 160 x 64
- Kích thước của filter : n x n x n_{c2} = 3 x 3 x 64
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 3 :

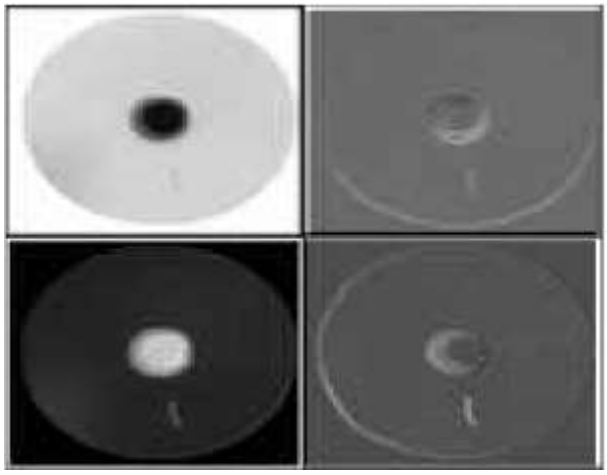
$$\left[\frac{w2+2p-n}{s} + 1\right] \text{x} \left[\frac{h2+2p-n}{s} + 1\right] \text{x k} = \left[\frac{160+2.1-3}{1} + 1\right] \text{x} \left[\frac{160+2.1-3}{1} + 1\right] \text{x 128}$$

$$w_3\text{'} \text{ x } h_3\text{'} \text{ x k} = 160 \text{ x } 160 \text{ x } 128 \quad (3.3)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Sau khi qua lớp MaxPooling, kích thước ảnh đầu ra sẽ là :

$$w_3 \text{ x } h_3 \text{ x k} = 80 \text{ x } 80 \text{ x } 128$$



Hình 3. 6 Một số đặc trưng trong 128 lớp

d. Lớp C2f (1) :

Sau khi qua khối Split, mỗi nhánh Bottleneck 1 và Concat đều nhận một nửa đặc trưng có kích thước là : w_3 x h_3 x $\frac{1}{2}n_{c3}$ = 80 x 80 x 64

Bottleneck 1 :

Các ma trận filter có kích thước 3 x 3 x 64 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck 1 :

- Số filter : k = 64
- Kích thước ảnh đầu vào : w_3 x h_3 x $\frac{1}{2}n_{c3}$= 80 x 80 x 64
- Kích thước của filter : n x n x $\frac{1}{2}n_{c3}$ = 3 x 3 x 64
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck 1 :

$$\left[\frac{w3+2p-n}{s} + 1\right] \text{ x } \left[\frac{h3+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{80+2.1-3}{1} + 1\right] \text{ x } \left[\frac{80+2.1-3}{1} + 1\right] \text{ x 64}$$

$$w_3\text{' x } h_3\text{' x k} = 80 \text{ x } 80 \text{ x } 64 \quad (3.4)$$

Bottleneck 2 : Quá trình tính toán tương tự Bottleneck 1

Đầu ra các khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : w_3 x h_3 x $\frac{3}{2}$ n_{c3} = 80 x 80 x 192

Các thông số ở Conv cuối cùng :

- Số filter : k = 128
- Kích thước ảnh đầu vào : w_3 x h_3 x $\frac{3}{2}n_{c3}$= 80 x 80 x 192
- Kích thước của filter : n x n x $\frac{3}{2}n_{c3}$ = 1 x 1 x 192
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w3+2p-n}{s} + 1\right] \text{ x } \left[\frac{h3+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{80+2.0-1}{1} + 1\right] \text{ x } \left[\frac{80+2.0-1}{1} + 1\right] \text{ x 128}$$

$$w_4 \text{ x } h_4 \text{ x k} = 80 \text{ x } 80 \text{ x } 128 \quad (3.5)$$

e. Lớp Conv (4) :

Các thông số ở lớp tích chập thứ 4 :

- Số filter : k = 256
- Kích thước ảnh đầu vào : w_4 x h_4 x n_{c4} = 80 x 80 x 128
- Kích thước của filter : n x n x n_{c4} = 3 x 3 x 128
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 4 :

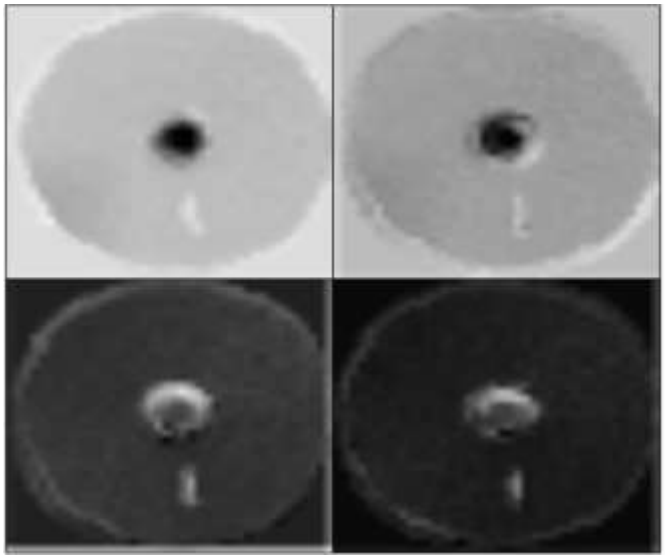
$$\left[\frac{w4+2p-n}{s} + 1\right] \text{x} \left[\frac{h4+2p-n}{s} + 1\right] \text{x k} = \left[\frac{80+2.1-3}{1} + 1\right] \text{x} \left[\frac{80+2.1-3}{1} + 1\right] \text{x 256}$$

$$w_5\text{'} \text{ x } h_5\text{'} \text{ x k} = 80 \text{ x } 80 \text{ x } 256 \quad (3.6)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Sau khi qua lớp MaxPooling, kích thước ảnh đầu ra sẽ là :

$$w_5 \text{ x } h_5 \text{ x k} = 40 \text{ x } 40 \text{ x } 256$$



Hình 3. 7 Một số đặc trưng trong 256 lớp

f. Lớp C2f (2) :

Sau khi qua khối Split, mỗi nhánh Bottleneck 1 và Concat đều nhận một nửa đặc trưng có kích thước là : w_5 x h_5 x $\frac{1}{2}n_{c5}$ = 40 x 40 x 128

Bottleneck 1 :

Các ma trận filter có kích thước 3 x 3 x 128 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck 1 :

- Số filter : k = 128
- Kích thước ảnh đầu vào : w_5 x h_5 x $\frac{1}{2}n_{c5}$= 40 x 40 x 128
- Kích thước của filter : n x n x $\frac{1}{2}n_{c5}$ = 3 x 3 x 128
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck 1 :

$$\left[\frac{w5+2p-n}{s} + 1\right] \text{ x } \left[\frac{h5+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{40+2.1-3}{1} + 1\right] \text{ x } \left[\frac{40+2.1-3}{1} + 1\right] \text{ x 128}$$

$$w_6\text{' x } h_6\text{' x k} = 40 \text{ x } 40 \text{ x } 128 \quad (3.7)$$

Bottleneck 2 : Quá trình tính toán tương tự Bottleneck 1

Đầu ra các khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : w_6' x h_6' x $\frac{3}{2}$ n_{c5} = 40 x 40 x 384

Các thông số ở Conv cuối cùng :

- Số filter : k = 256
- Kích thước ảnh đầu vào : w_6' x h_6' x $\frac{3}{2}$ n_{c5} = 40 x 40 x 384
- Kích thước của filter : n x n x $\frac{3}{2}n_{c5}$ = 1 x 1 x 384
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w6\prime+2p-n}{s} + 1\right] \text{ x } \left[\frac{h6\prime+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{40+2.0-1}{1} + 1\right] \text{ x } \left[\frac{40+2.0-1}{1} + 1\right] \text{ x 256}$$

$$w_6 \text{ x } h_6 \text{ x k} = 40 \text{ x } 40 \text{ x } 256 \quad (3.8)$$

g. Lớp Conv (5) :

Các thông số ở lớp tích chập thứ 5 :

- Số filter : k = 512
- Kích thước ảnh đầu vào : w_6 x h_6 x n_{c6} = 40 x 40 x 256
- Kích thước của filter : n x n x n_{c6} = 3 x 3 x 256
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 5 :

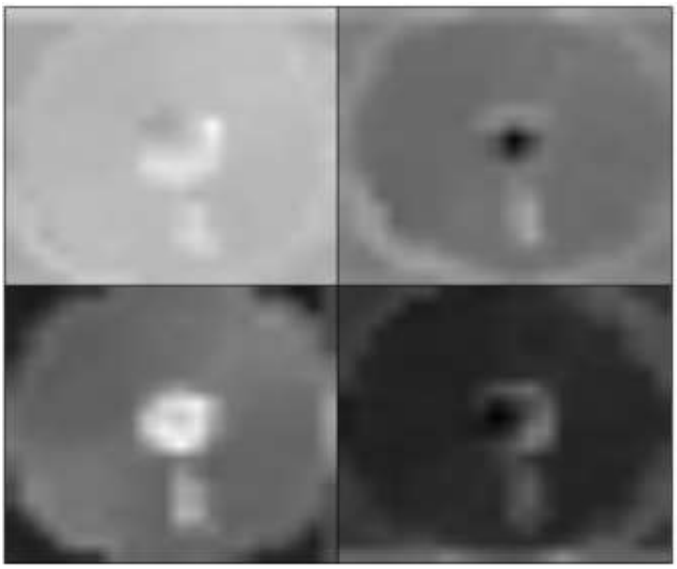
$$\left[\frac{w6+2p-n}{s} + 1\right] \text{ x } \left[\frac{h6+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{40+2.1-3}{1} + 1\right] \text{ x } \left[\frac{40+2.1-3}{1} + 1\right] \text{ x 512}$$

$$w_7\text{' x } h_7\text{' x k} = 40 \text{ x } 40 \text{ x } 512 \quad (3.9)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Sau khi qua lớp MaxPooling, kích thước ảnh đầu ra sẽ là :

$$w_7 \text{ x } h_7 \text{ x } k = 20 \text{ x } 20 \text{ x } 512$$



Hình 3. 8 Một số đặc trưng trong 512 lớp

h. Lớp C2f (3) :

Sau khi qua khối Split, mỗi nhánh Bottleneck và Concat đều nhận một nửa đặc trưng có kích thước là : $w_7 \text{ x } h_7 \text{ x } \frac{1}{2}n_{c7} = 20 \text{ x } 20 \text{ x } 256$

Bottleneck 1 :

Các ma trận filter có kích thước 3 x 3 x 256 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck 1 :

- Số filter : k = 256
- Kích thước ảnh đầu vào : $w_7 \text{ x } h_7 \text{ x } \frac{1}{2}n_{c7} = 40 \text{ x } 40 \text{ x } 256$
- Kích thước của filter : $n \text{ x } n \text{ x } \frac{1}{2}n_{c7} = 3 \text{ x } 3 \text{ x } 256$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck 1 :

$$\left[\frac{w7+2p-n}{s} + 1\right] \text{ x } \left[\frac{h7+2p-n}{s} + 1\right] \text{ x } k = \left[\frac{20+2.1-3}{1} + 1\right] \text{ x } \left[\frac{20+2.1-3}{1} + 1\right] \text{ x } 256$$

$$w_8' \text{ x } h_8' \text{ x } k = 20 \text{ x } 20 \text{ x } 256 \quad (3.10)$$

Bottleneck 2 : Quá trình tính toán tương tự Bottleneck 1

Đầu ra các khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : $w_8' \text{ x } h_8' \text{ x } \frac{3}{2} n_{c7} = 20 \text{ x } 20 \text{ x } 768$

Các thông số ở Conv cuối cùng :

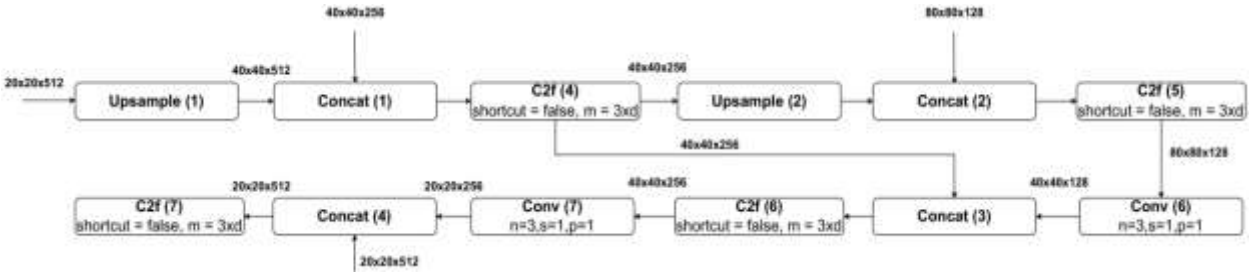
- Số filter : k = 512
- Kích thước ảnh đầu vào : $w_8' \text{ x } h_8' \text{ x } \frac{3}{2} n_{c7} = 20 \text{ x } 20 \text{ x } 768$
- Kích thước của filter : $n \text{ x } n \text{ x } \frac{3}{2} n_{c7} = 1 \text{ x } 1 \text{ x } 768$
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w7+2p-n}{s} + 1\right] \text{ x } \left[\frac{h7+2p-n}{s} + 1\right] \text{ x } k = \left[\frac{20+2.0-1}{1} + 1\right] \text{ x } \left[\frac{20+2.0-1}{1} + 1\right] \text{ x } 512$$

$$w_8 \text{ x } h_8 \text{ x } k = 20 \text{ x } 20 \text{ x } 512 \quad (3.11)$$

3.2.1.2. Phần Neck :



Hình 3. 9 Phần Neck

a. Khối Upsample (1) :

Đầu vào khối Upsample (1) nhận giá trị đầu vào từ khối C2f (3).

Kích thước ảnh đầu ra sau khi qua khối Upsample :

$$w_9 \text{ x } h_9 \text{ x } k = 40 \text{ x } 40 \text{ x } 512$$

b. Khối Concat (1) :

Khối Concat (1) này sẽ tổng hợp thông tin đặc trưng của khối C2f (2) và C2f(3).

Kích thước ma trận sau khi qua khối Concat (1) :

$w_{10} \times h_{10} \times k = (w_9 \times h_9 \times 512) + (w_6 \times h_6 \times 256) = 40 \times 40 \times 768$ (3.12)

c. Khối C2f (4) :

Sau khi qua khối Split,mỗi nhánh Bottleneck và Concat đều nhận một nửa đặc trưng có kích thước là : $w_{10} \times h_{10} \times \frac{1}{2}n_{c10} = 40 \times 40 \times 384$

Bottleneck :

Các ma trận filter có kích thước 3 x 3 x 384 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck :

- Số filter : k = 384
- Kích thước ảnh đầu vào : $w_{10} \times h_{10} \times \frac{1}{2}n_{c10} = 40 \times 40 \times 384$
- Kích thước của filter : $n \times n \times \frac{1}{2}n_{c10} = 3 \times 3 \times 384$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck :

$$\left[\frac{w10+2p-n}{s} + 1\right] \times \left[\frac{h10+2p-n}{s} + 1\right] \times k = \left[\frac{40+2.1-3}{1} + 1\right] \times \left[\frac{40+2.1-3}{1} + 1\right] \times 384$$

$$w_{11}' \times h_{11}' \times k = 40 \times 40 \times 384 \quad (3.13)$$

Đầu ra khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : $w_{11}' \times h_{11}' \times n_{c10} = 40 \times 40 \times 768$

Các thông số ở Conv cuối cùng :

- Số filter : k = 256
- Kích thước ảnh đầu vào : $w_{11}' \times h_{11}' \times n_{c10} = 40 \times 40 \times 768$
- Kích thước của filter : $n \times n \times n_{c10} = 1 \times 1 \times 768$
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w11'+2p-n}{s} + 1\right] \times \left[\frac{h11'+2p-n}{s} + 1\right] \times k = \left[\frac{40+2.0-1}{1} + 1\right] \times \left[\frac{40+2.0-1}{1} + 1\right] \times 256$$

$$w_{11} \times h_{11} \times k = 40 \times 40 \times 256 \quad (3.14)$$

d. Khối Upsample (2) :

Đầu vào khối Upsample (1) nhận giá trị đầu vào từ khối C2f (4).

Kích thước ảnh đầu ra sau khi qua khối Upsample :

$$w_{12} \text{ x } h_{12} \text{ x } k = 80 \text{ x } 80 \text{ x } 256$$

e. Khối Concat (2) :

Khối Concat (2) này sẽ tổng hợp thông tin đặc trưng của khối C2f (1) và C2f(4).

Kích thước ma trận sau khi qua khối Concat (2) :

$$w_{13} \text{ x } h_{13} \text{ x } k = (w_{12} \text{ x } h_{12} \text{ x } 256) + (w_4 \text{ x } h_4 \text{ x } 128) = 80 \text{ x } 80 \text{ x } 384 \qquad (3.15)$$

f. Khối C2f (5) :

Sau khi qua khối Split, mỗi nhánh Bottleneck và Concat đều nhận một nửa đặc trưng có kích thước là : $w_{13} \text{ x } h_{13} \text{ x } \frac{1}{2}n_{c13} = 80 \text{ x } 80 \text{ x } 192$

Bottleneck :

Các ma trận filter có kích thước 3 x 3 x 192 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck :

- Số filter : k = 192
- Kích thước ảnh đầu vào : $w_{13} \text{ x } h_{13} \text{ x } \frac{1}{2}n_{c13} = 80 \text{ x } 80 \text{ x } 192$
- Kích thước của filter : $n \text{ x } n \text{ x } \frac{1}{2}n_{c13} = 3 \text{ x } 3 \text{ x } 192$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck :

$$\left[\frac{w10+2p-n}{s} + 1\right] \text{x} \left[\frac{h10+2p-n}{s} + 1\right] \text{x } k = \left[\frac{80+2.1-3}{1} + 1\right] \text{x} \left[\frac{80+2.1-3}{1} + 1\right] \text{x } 192$$

$$w_{14}' \text{ x } h_{14}' \text{ x } k = 80 \text{ x } 80 \text{ x } 192 \qquad (3.16)$$

Đầu ra khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : $w_{14}' \text{ x } h_{14}' \text{ x } n_{c13} = 80 \text{ x } 80 \text{ x } 384$

Các thông số ở Conv cuối cùng :

- Số filter : k = 128
- Kích thước ảnh đầu vào : $w_{14}' \text{ x } h_{14}' \text{ x } n_{c13} = 80 \text{ x } 80 \text{ x } 384$
- Kích thước của filter : $n \text{ x } n \text{ x } n_{c10} = 1 \text{ x } 1 \text{ x } 384$
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w14\prime+2p-n}{s}+1\right] \text{x} \left[\frac{h14\prime+2p-n}{s}+1\right] \text{x k} = \left[\frac{80+2.0-1}{1}+1\right] \text{x} \left[\frac{80+2.0-1}{1}+1\right] \text{x } 128$$

$$w_{14} \text{ x } h_{14} \text{ x k} = 80 \text{ x } 80 \text{ x } 128 \quad (3.17)$$

Đầu ra khối C2f (5) sẽ đưa vào khối Conv (6) và được đưa vào khối Detect ở phần Head.

g. Khối Conv (6) :

Các thông số ở lớp tích chập thứ 6 :

- Số filter : k = 128
- Kích thước ảnh đầu vào : $w_{14} \text{ x } h_{14} \text{ x } n_{c14} = 80 \text{ x } 80 \text{ x } 128$
- Kích thước của filter : $n \text{ x } n \text{ x } n_{c14} = 3 \text{ x } 3 \text{ x } 128$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 6 :

$$\left[\frac{w14+2p-n}{s}+1\right] \text{x} \left[\frac{h14+2p-n}{s}+1\right] \text{x k} = \left[\frac{40+2.1-3}{1}+1\right] \text{x} \left[\frac{40+2.1-3}{1}+1\right] \text{x } 512$$

$$w_{15}\text{' x } h_{15}\text{' x k} = 40 \text{ x } 40 \text{ x } 512 \quad (3.18)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Tiếp theo ảnh sẽ được cho qua lớp MaxPooling giảm kích thước ảnh.

Kích thước ma trận sau khi qua khối Conv (6) :

$$w_{15} \text{ x } h_{15} \text{ x k} = 40 \text{ x } 40 \text{ x } 128$$

h. Khối Concat (3) :

Khối Concat (3) này sẽ tổng hợp thông tin đặc trưng của khối C2f (4) và C2f(5).

Kích thước ma trận sau khi qua khối Concat (3) :

$$w_{16} \text{ x } h_{16} \text{ x k} = (w_{15} \text{ x } h_{15} \text{ x } 128) + (w_{11} \text{ x } h_{11} \text{ x } 256) = 40 \text{ x } 40 \text{ x } 384 \quad (3.19)$$

i. Khối C2f (6) :

Sau khi qua khối Split, mỗi nhánh Bottleneck và Concat đều nhận một nửa đặc trưng có kích thước là : $w_{16} \text{ x } h_{16} \text{ x } \frac{1}{2} n_{c16} = 40 \text{ x } 40 \text{ x } 192$

Bottleneck :

Các ma trận filter có kích thước 3 x 3 x 192 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck :

- Số filter : k = 192
- Kích thước ảnh đầu vào : w_{16} x h_{16} x $\frac{1}{2}n_{c16}$ = 40 x 40 x 192
- Kích thước của filter : n x n x $\frac{1}{2}n_{c16}$ = 3 x 3 x 192
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck :

$$\left[\frac{w16+2p-n}{s} + 1\right] \text{ x } \left[\frac{h16+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{40+2.1-3}{1} + 1\right] \text{ x } \left[\frac{40+2.1-3}{1} + 1\right] \text{ x 192}$$

$$w_{17}\text{' x } h_{17}\text{' x k} = 40 \text{ x } 40 \text{ x } 192 \quad (3.20)$$

Đầu ra khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : w_{17}' x h_{17}' x n_{c16} = 40 x 40 x 384

Các thông số ở Conv cuối cùng :

- Số filter : k = 256
- Kích thước ảnh đầu vào : w_{17}' x h_{17}' x n_{c16} = 40 x 40 x 384
- Kích thước của filter : n x n x n_{c16} = 1 x 1 x 384
- Padding và Stride : p = 0, s =1

Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w17\prime+2p-n}{s} + 1\right] \text{ x } \left[\frac{h17\prime+2p-n}{s} + 1\right] \text{ x k} = \left[\frac{40+2.0-1}{1} + 1\right] \text{ x } \left[\frac{40+2.0-1}{1} + 1\right] \text{ x 256}$$

$$w_{14} \text{ x } h_{14} \text{ x k} = 40 \text{ x } 40 \text{ x } 256 \quad (3.21)$$

Đầu ra khối C2f (6) sẽ đưa vào khối Conv (7) và được đưa vào khối Detect ở phần Head.

j. Khối Conv (7) :

Các thông số ở lớp tích chập thứ 6 :

- Số filter : k = 128
- Kích thước ảnh đầu vào : w_{17} x h_{17} x n_{c17} = 80 x 80 x 128
- Kích thước của filter : n x n x n_{c17} = 3 x 3 x 128
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp tích chập thứ 7 :

$$\left[\frac{w17+2p-n}{s}+1\right] x7\left[\frac{h17+2p-n}{s}+1\right] x\ k = \left[\frac{40+2.1-3}{1}+1\right] x \left[\frac{40+2.1-3}{1}+1\right] x\ 512$$

$$w_{18}' \ x\ h_{18}' \ x\ k = 40 \ x\ 40 \ x\ 512 \qquad (3.22)$$

Sau đó ma trận ảnh sẽ tiếp tục được đưa vào hàm kích hoạt Silu.

Tiếp theo ảnh sẽ được cho qua lớp MaxPooling giảm kích thước ảnh.

Kích thước ma trận sau khi qua khối Conv (7) :

$$w_{18} \ x\ h_{18} \ x\ k = 20 \ x\ 20 \ x\ 256$$

k. Khối Concat (4) :

Khối Concat (4) này sẽ tổng hợp thông tin đặc trưng của khối C2f (3) và C2f(6).

Kích thước ma trận sau khi qua khối Concat (4) :

$$w_{19} \ x\ h_{19} \ x\ k = (w_{18} \ x\ h_{18} \ x\ 256) + (w_8 \ x\ h_8 \ x\ 512) = 40 \ x\ 40 \ x\ 768 \qquad (3.23)$$

l. Khối C2f (7) :

Sau khi qua khối Split, mỗi nhánh Bottleneck và Concat đều nhận một nửa đặc trưng có kích thước là : $w_{19} \ x\ h_{19} \ x\ \frac{1}{2}n_{c19} = 20 \ x\ 20 \ x\ 384$

Bottleneck :

Các ma trận filter có kích thước 3 x 3 x 384 có các trọng số là ẩn lần lượt là w_{11}, w_{12},.. sẽ được tinh chỉnh mỗi lần mô hình học để làm giảm loss dựa trên số epochs đưa ra.

Các thông số ở Bottleneck :

- Số filter : k = 384
- Kích thước ảnh đầu vào : $w_{19} \ x\ h_{19} \ x\ \frac{1}{2}n_{c19} = 20 \ x\ 20 \ x\ 384$
- Kích thước của filter : $n \ x\ n \ x\ \frac{1}{2}n_{c19} = 3 \ x\ 3 \ x\ 384$
- Padding và Stride : p = 1, s =1

Kích thước ma trận mới được tạo ra sau khi qua lớp Bottleneck :

$$\left[\frac{w19+2p-n}{s}+1\right] x \left[\frac{h19+2p-n}{s}+1\right] x\ k = \left[\frac{20+2.1-3}{1}+1\right] x \left[\frac{20+2.1-3}{1}+1\right] x\ 384$$

$$w_{20}' \ x\ h_{20}' \ x\ k = 20 \ x\ 20 \ x\ 384 \qquad (3.24)$$

Đầu ra khối Bottleneck sẽ được đưa vào khối Concat.

Kích thước ảnh sau khi qua khối Concat : w_{20}' x h_{20}' x n_{c19}= 20 x 20 x 768

Các thông số ở Conv cuối cùng :

- Số filter : k = 512
- Kích thước ảnh đầu vào : w_{20}' x h_{20}' x n_{c19}= 20 x 20 x 768
- Kích thước của filter : n x n x n_{c10} = 1 x 1 x 768
- Padding và Stride : p = 0, s =1

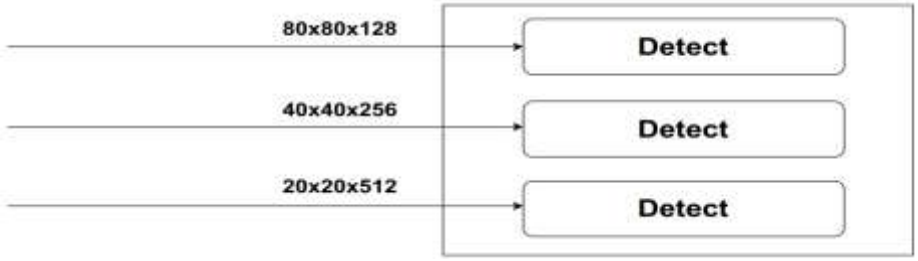
Kích thước ma trận mới được tạo ra sau khi qua Conv cuối cùng :

$$\left[\frac{w20\prime+2p-n}{s}+1\right] \text{ x } \left[\frac{h20\prime+2p-n}{s}+1\right] \text{ x k} = \left[\frac{20+2.0-1}{1}+1\right] \text{ x } \left[\frac{20+2.0-1}{1}+1\right] \text{ x 512}$$

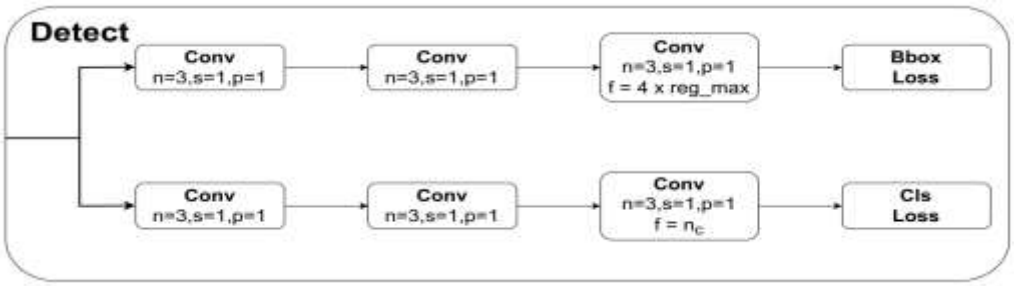
$$w_{20} \text{ x } h_{20} \text{ x k} = 20 \text{ x } 20 \text{ x } 512 \quad (3.25)$$

Đầu ra khối C2f (7) sẽ được đưa vào khối Detect ở phần Head.

3.2.1.3. Phần Head :



Hình 3. 10 Phần Head



Hình 3. 11 Khối Detect

a. Dự đoán toạ độ hộp (Bbox Loss) :

Vì sử dụng YoLov8n nên reg_max = 4, mỗi toạ độ (left, top, right, bottom) là 1 vector có độ dài là phân phối xác suất rời rạc trên các lớp số nguyên từ 0 đến reg_max-1. Từ đó, có được 4 x reg_max đầu ra. Dùng hàm softmax để biến vector thành hàm phân phối xác suất.

Reg_max = 4 nên chia các toạ độ thành các bin [0, 1, 2, 3]

Số lượng các thông số đầu ra : f = 4 x reg_max = 4 x 4 = 16

Cách sắp xếp đầu ra :

- 16 các giá trị cho 4 toạ độ -> mỗi toạ độ có 4 giá trị rời rạc.
- Chia 16 giá trị này thành 4 nhóm :
  - Nhóm 1 : toạ độ trái (left)
  - Nhóm 2 : toạ độ trên (top)
  - Nhóm 3 : toạ độ phải (right)
  - Nhóm 4 : toạ độ dưới (botttom)

$$\text{f = 4 x Reg\_max} = \begin{bmatrix} l_0 & l_1 & l_2 & l_3 \\ t_0 & t_1 & t_2 & t_3 \\ r_0 & r_1 & r_2 & r_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix}$$

Xét toạ độ của Bbox :

Giả sử giá trị ban đầu chưa qua huấn luyện là :

$$\text{f} = \begin{bmatrix} 2.1 & 0.9 & 1.2 & 0.0 \\ 0.1 & 0.3 & 2.5 & 1.1 \\ 1.5 & 1.0 & 0.5 & 0.2 \\ 0.0 & 1.0 & 1.5 & 0.5 \end{bmatrix}$$

Với L = [2.1 0.9 1.2 0.0]

T = [0.1 0.3 2.5 1.1]

R = [1.5 1.0 0.5 0.2]

B = [0.0 1.0 1.5 0.5]

Tiến hành đưa qua hàm kích hoạt softmax để cho ra 16 phân phối xác suất rời rạc.

$$\text{P(L)} = \text{Softmax(L)} = \frac{e^{L_i}}{\sum_{j=1}^{n} e^{L_j}} \text{ với i = 1, 2,…, n} \quad (3.26)$$

$\sum_{j=1}^{n} e^{L_j} = e^{L_0} + e^{L_1} + e^{L_2} + e^{L_3} = e^{2.1} + e^{0.9} + e^{1.2} + e^{0.0} = 14.95$

$\text{P}(L_0) = \frac{e^{L_0}}{14.95} = \frac{e^{2.1}}{14.95} = 0.55$, $\text{P}(L_1) = \frac{e^{L_1}}{14.95} = \frac{e^{0.9}}{14.95} = 0.17$

$\text{P}(L_2) = \frac{e^{L_2}}{14.95} = \frac{e^{1.2}}{14.95} = 0.22$, $\text{P}(L_3) = \frac{e^{L_3}}{14.95} = \frac{e^{0}}{14.95} = 0.07$

⇨ P(L) = [0.55 0.17 0.22 0.07]

$$P(T) = Softmax(T) = \frac{e^{T_i}}{\sum_{j=1}^{n} e^{T_j}} \text{ với i = 1, 2,…, n} \quad (3.27)$$

$\sum_{j=1}^{n} e^{T_j} = e^{T_0} + e^{T_1} + e^{T_2} + e^{T_3} = e^{0.1} + e^{0.3} + e^{2.5} + e^{1.1} = 17.64$

$P(T_0) = \frac{e^{T_0}}{17.64} = \frac{e^{0.1}}{17.64} = 0.06$, $P(T_1) = \frac{e^{T_1}}{17.64} = \frac{e^{0.3}}{17.64} = 0.08$

$P(T_2) = \frac{e^{T_2}}{17.64} = \frac{e^{2.5}}{17.64} = 0.69$, $P(T_3) = \frac{e^{T_3}}{17.64} = \frac{e^{1.1}}{17.64} = 0.17$

⇨ P(T) = [0.06 0.08 0.69 0.17]

$$P(R) = Softmax(R) = \frac{e^{R_i}}{\sum_{j=1}^{n} e^{R_j}} \text{ với i = 1, 2,…, n} \quad (3.28)$$

$\sum_{j=1}^{n} e^{R_j} = e^{R_0} + e^{R_1} + e^{R_2} + e^{R_3} = e^{1.5} + e^{1.0} + e^{0.5} + e^{0.2} = 10.07$

$P(R_0) = \frac{e^{R_0}}{10.07} = \frac{e^{1.5}}{10.07} = 0.45$, $P(R_1) = \frac{e^{R_1}}{10.07} = \frac{e^{1.0}}{10.07} = 0.27$

$P(R_2) = \frac{e^{R_2}}{10.07} = \frac{e^{0.5}}{10.07} = 0.16$, $P(R_3) = \frac{e^{R_3}}{10.07} = \frac{e^{0.2}}{10.07} = 0.12$

⇨ P(R) = [0.45 0.27 0.16 0.12]

$$P(B) = Softmax(B) = \frac{e^{R_i}}{\sum_{j=1}^{n} e^{R_j}} \text{ với i = 1, 2,…, n} \quad (3.29)$$

$\sum_{j=1}^{n} e^{B_j} = e^{B_0} + e^{B_1} + e^{B_2} + e^{B_3} = e^{0.0} + e^{1.0} + e^{1.5} + e^{0.5} = 9.85$

$P(B_0) = \frac{e^{B_0}}{9.85} = \frac{e^{0.0}}{9.85} = 0.1$, $P(B_1) = \frac{e^{B_1}}{9.85} = \frac{e^{1.0}}{9.85} = 0.27$

$P(B_2) = \frac{e^{B_2}}{9.85} = \frac{e^{1.5}}{9.85} = 0.45$, $P(B_3) = \frac{e^{B_3}}{9.85} = \frac{e^{0.5}}{9.85} = 0.17$

⇨ P(B) = [0.1 0.27 0.45 0.17]

Expected Value (giá trị mong đợi) :

$$\begin{aligned} d_{left} &= \sum_{i=0}^{3} i.P(L) = 0x0.55 + 1x0.17 + 2x0.22 + 3x0.07 = 0.82 \\ d_{top} &= \sum_{i=0}^{3} i.P(T) = 0x0.06 + 1x0.08 + 2x0.69 + 3x0.17 = 1.97 \\ d_{right} &= \sum_{i=0}^{3} i.P(R) = 0x0.45 + 1x0.27 + 2x0.16 + 3x0.12 = 0.95 \\ d_{bottom} &= \sum_{i=0}^{3} i.P(B) = 0x0.1 + 1x0.27 + 2x0.45 + 3x0.17 = 1.68 \end{aligned} \quad (3.30)$$

Trong đó :

d_{left}, d_{top}, d_{right}, d_{right} lần lượt là tọa độ cạnh trái, trên, phải và dưới của hộp

⇨ Toạ độ Bouding box dự đoán : $d_{left} = 0.82$, $d_{top} = 1.97$, $d_{right} = 0.95$, $d_{bottom} = 1.68$

Ground Truth (GT) là toạ độ lỗi thực tế của *hình 3.1* :

$$x = 0.57, y = 0.75, w = 0.09, h = 0.24$$

$$\begin{aligned} d_{left\_gt} &= x - \frac{1}{2}.w = 0.57 - \frac{1}{2}x0.09 = 0.525 \\ d_{right\_gt} &= x + \frac{1}{2}.w = 0.57 + \frac{1}{2}x0.09 = 0.615 \\ d_{top\_gt} &= y + \frac{1}{2}.h = 0.75 + \frac{1}{2}x0.24 = 0.87 \\ d_{bottom\_gt} &= y - \frac{1}{2}.h = 0.75 - \frac{1}{2}x0.24 = 0.63 \end{aligned} \quad (3.31)$$

Tiến hành phân rã giá trị trên thành giá trị rời rạc gần nhất :

Xét $d_{left\_gt} = 0.525$

Vì $d_{left\_gt} = 0.525$ nằm giữa bin 0 và bin 1, chia như sau :

- Ở vị trí bin 1 : 0.525 – 0 = 0.525
- Ở vị trí bin 0 : 1 – 0.525 = 0.475

⇨ Phân phối GT : [0.475 0.525 0 0]

Độ lệch giữa phân phối thực (GT) và phân phối dự đoán :

$$\mathcal{L}_{DFL}(L) = -\sum_{i=0}^{4} GT_L(i).log(P_L(i)) \quad (3.32)$$

$GT_L(i)$, $P_L(i)$ lần lượt là trọng số phân phối thực và xác xuất mô hình dự đoán tại i.

⇨ $\mathcal{L}_{DFL}(L) = -(0.475x\log(0.55) + 0.525x\log(0.17)) = 0.53$

Xét $d_{right\_gt} = 0.615$

Vì $d_{right\_gt} = 0.615$ nằm giữa bin 0 và bin 1, chia như sau :

- Ở vị trí bin 1 : 0.615 – 0 = 0.615
- Ở vị trí bin 0 : 1 – 0.615 = 0.385

⇨ Phân phối GT : [0.385 0.615 0 0]

Độ lệch giữa phân phối thực (GT) và phân phối dự đoán :

$$\mathcal{L}_{DFL}(R) = -\sum_{i=0}^{4} GT_R(i).log(P_R(i)) \quad (3.33)$$

$GT_R(i)$, $P_R(i)$ lần lượt là trọng số phân phối thực và xác xuất mô hình dự đoán tại i.

⇨ $\mathcal{L}_{DFL}(R) = -(0.385x\log(0.45) + 0.615x\log(0.2)) = 0.56$

Xét $d_{top\_gt} = 0.87$

Vì d_{top_gt} = 0.87 nằm giữa bin 0 và bin 1, chia như sau :

- Ở vị trí bin 1 : 0.87 – 0 = 0.87
- Ở vị trí bin 0 : 1 – 0.87 = 0.13

⇨ Phân phối GT : [0.13 0.87 0 0]

Độ lệch giữa phân phối thực (GT) và phân phối dự đoán :

$$\mathcal{L}_{DFL}(T) = -\sum_{i=0}^{4} GT_T(i).log(P_T(i)) \quad (3.34)$$

$GT_T(i)$, $P_T(i)$ lần lượt là trọng số phân phối thực và xác xuất mô hình dự đoán tại i.

⇨ $\mathcal{L}_{DFL}(T)$ = -(0.13xlog(0.06) + 0.87xlog(0.08)) = 1.11

Xét d_{bottom_gt} = 0.63

Vì d_{bottom_gt} = 0.63 nằm giữa bin 0 và bin 1, chia như sau :

- Ở vị trí bin 1 : 0.63– 0 = 0.63
- Ở vị trí bin 0 : 1 – 0.63 = 0.37

⇨ Phân phối GT : [0.37 0.63 0 0]

Độ lệch giữa phân phối thực (GT) và phân phối dự đoán :

$$\mathcal{L}_{DFL}(B) = -\sum_{i=0}^{4} GT_B(i).log(P_B(i)) \quad (3.35)$$

$GT_B(i)$, $P_B(i)$ lần lượt là trọng số phân phối thực và xác xuất mô hình dự đoán tại i.

⇨ $\mathcal{L}_{DFL}(B)$ = -(0.37xlog(0.1) + 0.63xlog(0.27)) = 0,7

$$\mathcal{L}_{DFL} = \mathcal{L}_{DFL}(L) + \mathcal{L}_{DFL}(R) + \mathcal{L}_{DFL}(T) + \mathcal{L}_{DFL}(B) = 0.53 + 0.56 + 1.11 + 0.7 = 2.9 \quad (3.36)$$

b. Dự đoán lớp đối tượng (Cls Loss) :

Đầu vào ảnh sau khi qua 2 lớp Conv có kích thước 20x20x512 nên có 400 điểm, mỗi điểm gồm 512 đặc trưng. Tại điểm (i,j) vector đặc trưng là $f_{i,j}$ gồm 512 đặc trưng.

Lớp Conv cuối có Kernel 1x1, có số filter = n_c

k = 512, n_c = 2 (2 lớp là lỗi TXNL và STP)

Tại điểm (0,0), trích 6 đặc trưng từ 512 đặc trưng để tính toán, ta có :

$$f_{0,0} = [0.3, 0.1, -0.2. 0.5, 0.0, 0.4]$$

Lớp TXNL (class 0) : w_0 = [1.2, -0.5, 0.8, 0.6, 0.0, −0.3], b_0 = 0.9

Lớp STP (class 1) : w_1 = [−0.7, 0.9, −0.6, 1.1, 0.2, 0.5], b_1 = 0.6

$$z_{0,0} = [z_0, z_1]$$

$$z_0 = w_0 \text{ x } f_{0,0} + b_0 = 1.2 \text{ x } 0.3 + (-0.5) \text{ x } 0.1 + 0.8 \text{ x } (-0.2) + 0.6 \text{ x } 0.5 + 0.0 \text{ x } 0.0 + (-0.3) \text{ x } 0.4 + 0.9 \quad (3.37)$$
$$= 0.36 - 0.05 - 0.16 + 0.30 + 0.00 - 0.12 + 0.9 = 1.23$$

$$z_1 = w_1 \text{ x } f_{0,0} + b_1 = (-0.7) \text{ x } 0.3 + 0.9 \text{ x } 0.1 + (-0.6) \text{ x } (-0.2) + 1.1 \text{ x } 0.5 + 0.2 \text{ x } 0.0 + 0.5 \text{ x } 0.4 + 0.6 \quad (3.38)$$
$$= (-0.21) + 0.0\,9 + 0.12 + 0.55 + 0.00 + 0.20 + 0.6 = 1.35$$

⇨ $z_{0,0} = [1.23, 1.35]$

Tiến hành đưa qua hàm kích hoạt softmax để cho ra phân phối xác suất.

$$p_i = \frac{e^{z_i}}{\sum_j^n e^{z_i}} \text{ với i = 0,1,…n}$$

$$p_0 = \frac{e^{1.23}}{e^{1.23}+e^{1.35}} = 0.47$$

$$p_1 = \frac{e^{1.35}}{e^{1.23}+e^{1.35}} = 0.53$$

⇨ Mô hình dự đoán lỗi STP 53%

Hàm mất mát giữa giá trị dự đoán và giá trị thực tế :

$$BCE(p,y) = -[y \text{ x } \log(p) + (1-y) \text{ x } \log(1-p)] \quad (3.39)$$

Trong đó p và y lần là xác suất dự đoán mô hình và giá trị nhãn thật ($y_0 = 0$, $y_1 = 1$)

Với lỗi TXNL (class 0) :

$$Loss_0 = -[y_0 \text{ x } \log(p_0)+(1-y_0) \text{ x } \log(1-p_0)] = -[0 \text{ x } \log(0.47)+(1-0) \text{ x } \log(1-0.47)] = 0.276$$

Với lỗi STP (class 1) :

$$Loss_1 = -[y_1 \text{ x } \log(p_1)+(1-y_1) \text{ x } \log(1-p_1)] = -[1 \text{ x } \log(0.53)+(1-1) \text{ x } \log(1-0.53)] = 0.276$$

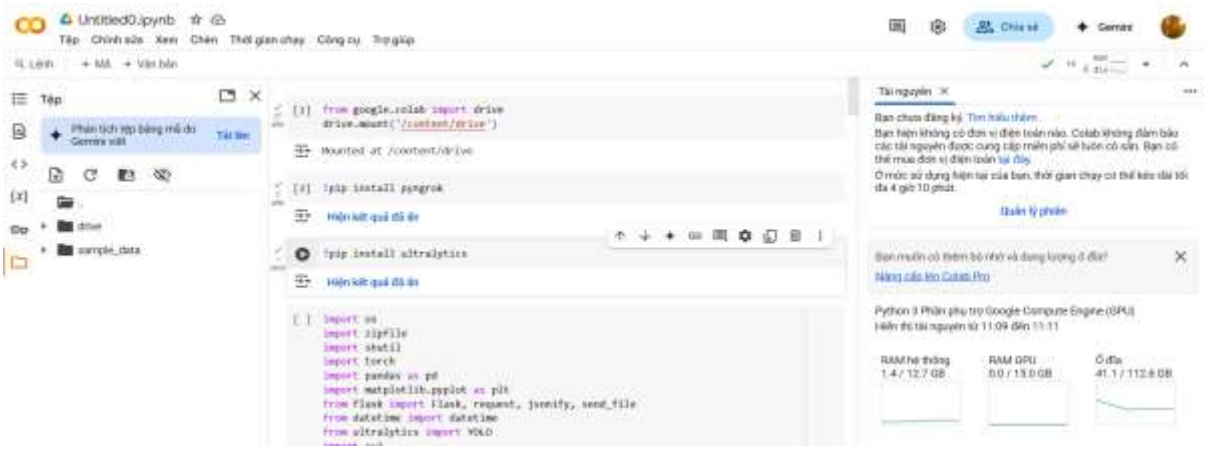
$$Cls\_loss = Loss_0 + Loss_1 = 0.276 + 0.276 = 0.552 \quad (3.40)$$

3.3. Thiết kế và triển khai mô hình :

3.3.1. Tạo flask API và giao diện websever upload dữ liệu từ người dùng :

a. Truy cập vào Google Colab :

Tiến hành đăng nhập vào Google Colab.



Hình 3. 12 Giao diện Google Colab

Lựa chọn thời gian chạy có GPU để học mô hình nhanh hơn và tiết kiệm thời gian.

b. Kết nối với Google Drive :

Vì sử dụng bản miễn phí nên Colab sẽ reset toàn bộ dữ liệu sau vài giờ sử dụng nên phải kết nối với GG drive để dữ liệu được lưu trữ vào GG drive mỗi khi Colab reset dữ liệu.

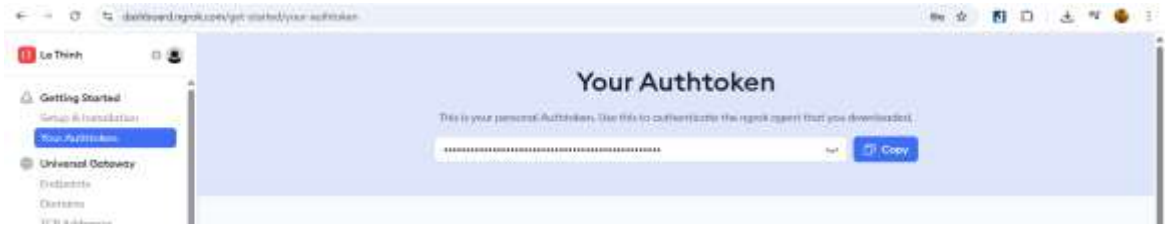
Viết dòng lệnh để kết nối từ Colab qua Drive :

```
from google.colab import drive
drive.mount('/content/drive')
```

c. Cấu hình Ngrok :

Khi flask API đưa ra đường link thì người dùng khi truy cập vào link đó sẽ không được vì chỉ là link local nên cần Ngrok để public link lên internet để mọi người truy cập được.

Đầu tiên sẽ truy cập vào web Ngrok, đăng kí tài khoản (nếu chưa có) và tiến hành đăng nhập để lấy key Authtoken.



Hình 3. 13 Trang web Ngrok để lấy Authtoken

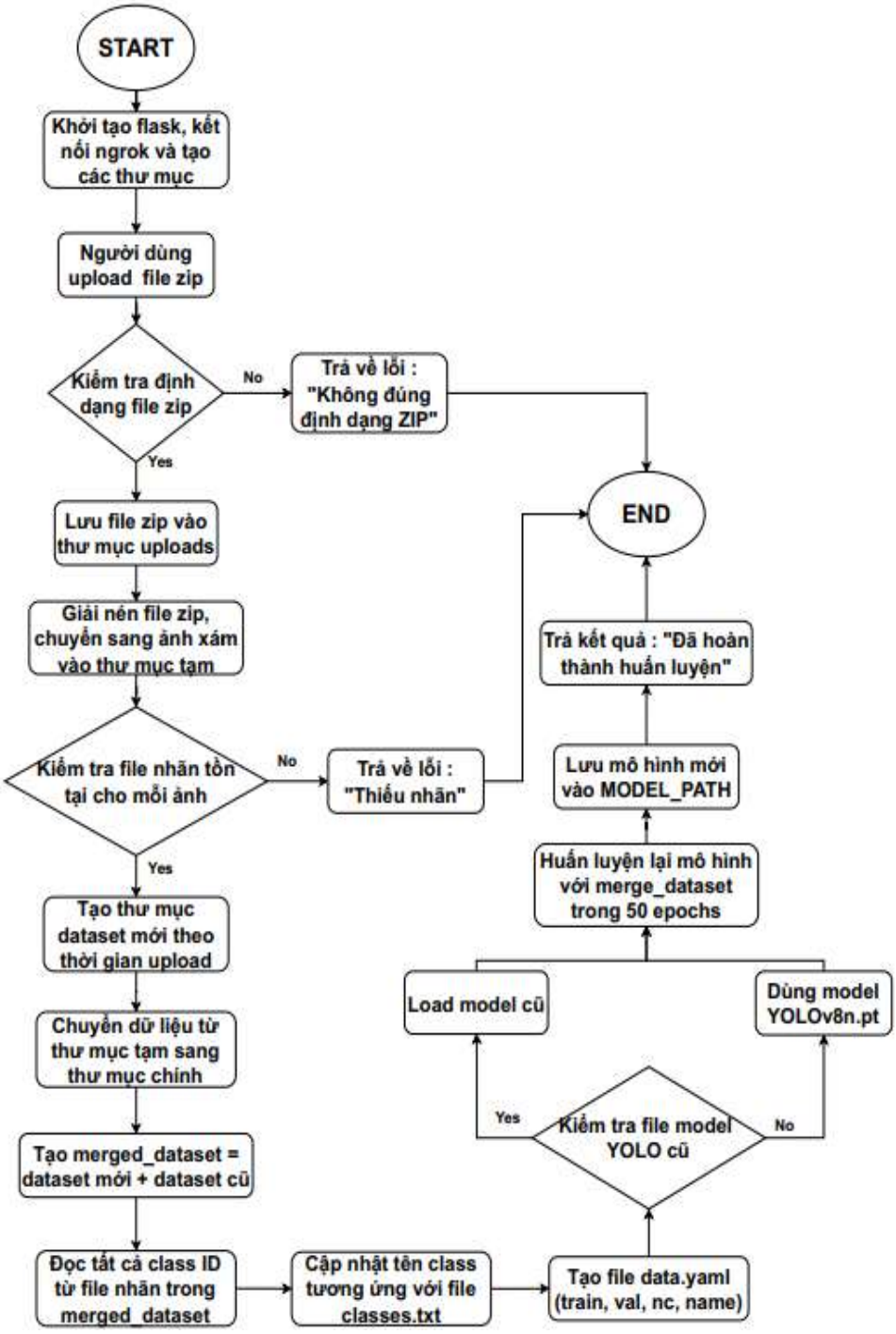
Key Authtoken lấy được từ trang chủ Ngrok sẽ gắn vào ô **giá trị** bên GG Colab.

d. Viết chương trình tạo websever và huấn luyện dữ liệu :

Để chương trình flask chạy được trên GG Coalab cần phải tải thêm các thư viện :

- Pyngrok : Để chương trình gọi hàm ngrok.
- Ultralytics : Để huấn luyện mô hình YoLo.

Lưu đồ thuật toán :



Hình 3. 14 Lưu đồ thuật toán cho chương trình huấn luyện mô hình

Tiến hành viết chương trình :

(Dưới đây chỉ trích ra để giải thích những đoạn chương trình chính, toàn bộ chương trình sẽ được đưa vào phần phụ lục).

❖ Import thư viện :

```
import os, zipfile, shutil, torch, pandas as pd, matplotlib.pyplot as plt
from flask import Flask, request, jsonify, send_file
from datetime import datetime
from ultralytics import YOLO
import cv2, threading, time, yaml
from pyngrok import ngrok, conf
from google.colab import userdata
```

- Giải thích :
  - Os, shutil, zipfile : Xử lý tệp/thư mục và giải nén ZIP.
  - Torch : Cho YOLOv8 hoạt động.
  - Pandas, matplotlib.pyplot : Dùng để log và vẽ biểu đồ (bạn muốn loại bỏ).
  - Flask: Tạo server xử lý HTTP request.
  - Datetime: Tạo timestamp để lưu log, tên folder...
  - Ultralytics.YOLO: Gọi model YOLOv8.
  - Cv2: OpenCV.
  - Threading: Chạy Flask trong luồng riêng khi dùng trong Colab.
  - Pyngrok: Tạo public URL qua ngrok.
  - Userdata: Lấy token ngrok từ người dùng Colab.

❖ Khởi tạo Flask + Ngrok :

```
app = Flask(__name__)
conf.get_default().auth_token = userdata.get("ngrok_token")
public_url = ngrok.connect(5000)
print("My tunnel =", public_url)
```

- Giải thích :
  - Tạo link web.
  - Dùng token ngrok để mở cổng 5000 ra public.

❖ Đường dẫn để lưu trữ dữ liệu :

```
UPLOAD_FOLDER = '/content/drive/MyDrive/data/uploads'
DATASET_FOLDER = '/content/drive/MyDrive/data/dataset'
TEMP_DATASET_FOLDER = '/content/drive/MyDrive/data/temp_dataset'
MERGED_FOLDER = '/content/drive/MyDrive/data/merged_dataset'
MODEL_PATH = '/content/drive/MyDrive/data/model/best.pt'
```

- Các đường dẫn đều đưa file tạo ra vào tài khoản GG Drive.
- Giải thích :
  - UPLOAD_FOLDER: Lưu file ZIP upload.
  - TEMP_DATASET_FOLDER: Giải nén tạm thời để kiểm tra.
  - MERGED_FOLDER: Dataset đã gộp nhiều lần upload.
  - MODEL_PATH: Đường dẫn file model YOLOv8.

❖ Tạo thư mục nếu chưa có :

```
os.makedirs(…, exist_ok=True)
```

- Giải thích : Tự động tạo thư mục vào GG Drive để lưu trữ.

❖ Hàm giải nén ZIP :

```
def extract_zip(zip_path, dest_folder):
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(dest_folder)
```

- Giải thích : Tạo hàm để giải nén file zip được upload lên từ công ty.

❖ Hàm chuyển ảnh RGB sang ảnh xám :

```
def convert_images_to_grayscale(root_folder):
…
```

❖ Xây dựng giao diện web để công ty upload zip chứa dataset :

```
@app.route('/')
def index():
    return '''
    <h2>Upload ZIP chứa ảnh, nhãn và tên lỗi </h2>
    …
    '''
```

- Giải thích : Mã HTML tạo ra giao diện trên websever để upload dataset

Hình 3. 15 Giao diện uploads dữ liệu huấn luyện

❖ Xử lý file ZIP data khi công ty upload lên :

```
@app.route('/upload_dataset', methods=['POST'])
def upload_dataset():
  …
```

- Giải thích : Trong đoạn chương trình trên sẽ có các phần chính như sau :
  - Lưu file ZIP.
  - Giải nén ZIP.
  - Đọc file Classes.txt.
  - Kiểm tra nhãn đã đầy đủ với mỗi ảnh chưa.
  - Tạo thư mục chứa dataset theo thời gian (timestamp).
  - Gộp toàn bộ dữ liệu mới và cũ vào **MERGED_FOLDER**.
  - Tạo file **data.yaml** để train YOLO.
  - Load YOLOv8n.pt.
  - Tiến hành train mô hình.
  - Lưu model mới vào GG Drive.

❖ Khởi chạy Flask sever

```
def run_flask():
  app.run(debug=False, port=5000, use_reloader=False)
if __name__ == '__main__':
  threading.Thread(target=run_flask).start()
  print("Flask server is running...")
  while True:
    time.sleep(1000)
```

- Giải thích : Đoạn chương trình trên sẽ đảm bảo Flask chạy liên tục trong GG Colab mà không bị ngắt giữa chừng.

3.3.2. Xây dựng bộ dữ liệu để gửi zip lên cho mô hình tự học :

Để upload bộ dữ liệu cho mô hình học tốt thì cần nhiều ảnh để mô hình không bị quên khi học các lỗi mới.

a. Chuẩn bị bộ dữ liệu ảnh :

Đầu tiên ảnh cuộn sợi lỗi được chụp lại từ thực tế.

Vì số lượng ảnh lỗi thu thập được từ thực tế quá ít nên sẽ chụp những ảnh cuộn sợi tốt ở nhiều góc, không lỗi đề photoshop lỗi lên, tăng cường thêm nhiều dữ liệu ảnh hơn.

Hình 3. 16 Cuộn sợi được chụp từ thực tế

Tiến hành xoá phông nền để không bị xen lẫn các vật thể khác giống hình lỗi với lỗi trên cuộn sợi và photoshop các lỗi lên cuộn sợi ở nhiều góc độ và nhiều vị trí lỗi cho ra khoảng 100 – 200 ảnh lỗi.

Hình 3. 17 Lỗi trên các cuộn sợi từ thực tế

b. Chuẩn bị bộ dữ liệu nhãn :

Mỗi ảnh lỗi sẽ phải đi kèm với nhãn có gắn toạ độ lỗi theo YOLO.

Để tạo file nhãn, tiến hành vào phần mềm pycharm, gõ dòng lệnh **labelImg** sẽ xuất giao diện tạo khung lỗi YOLO để đưa ra toạ độ lỗi.

Hình 3. 18 Giao diện gắn nhãn lỗi

Sau khi khoanh vùng toạ độ lỗi, lưu lại sẽ đưa ra file chưa toạ độ nhãn YOLO với các toạ độ :

Class | x | y | w | h

Trong đó :

- Class sẽ là lớp được file **data.yaml** hiểu là lỗi nào. (0,1,…)
- x là toạ độ vị trí trục x theo tỉ lệ ảnh.
- y là toạ độ vị trí trục y theo tỉ lệ ảnh.
- w là chiều rộng của vùng khoanh.
- h là chiều cao của vùng khoanh.

(Ví dụ toạ độ lỗi của ảnh trên : 0 0.418945 0.699219 0.115234 0.347656).

c. Chuẩn bị file ZIP chứa data để upload :

Sau khi chuẩn bị xong file chứa ảnh và file chứa nhãn thì tiến hành chia file ảnh và file nhãn thành 2 phần chính :

- File training set (train), 80% dữ liệu :
  - Dữ liệu để huấn luyện mô hình.
  - Mô hình sẽ học được từ các ảnh và nhãn trong file để hiểu rõ được đối tượng cần nhận diện.
- File validation set (val), 20% dữ liệu :
  - Dữ liệu để đánh giá chất lượng mô hình trong quá trình huấn luyện.
  - Mô hình không được học từ tập này, mà chỉ kiểm tra sau huấn luyện xem mô hình học tốt chưa.

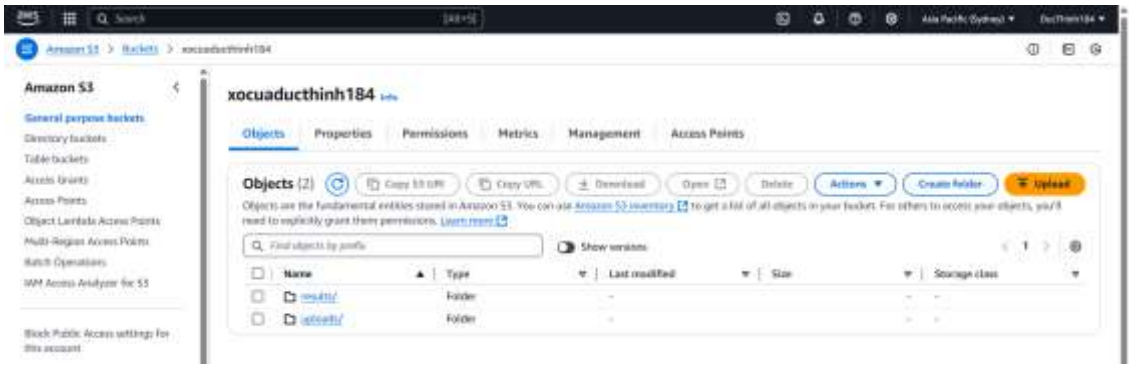
Cuối cùng là tạo 1 file Classes.txt để chứa tên lỗi.

3.3.3. Xây dựng Cloud AWS để deploy model và nhận diện ảnh từ người dùng :

a. Lựa chọn các dịch vụ cần thiết cho mô hình :

❖ Chọn S3 và thiết lập để làm nơi lưu trữ dữ liệu ảnh người dùng upload và ảnh kết quả.

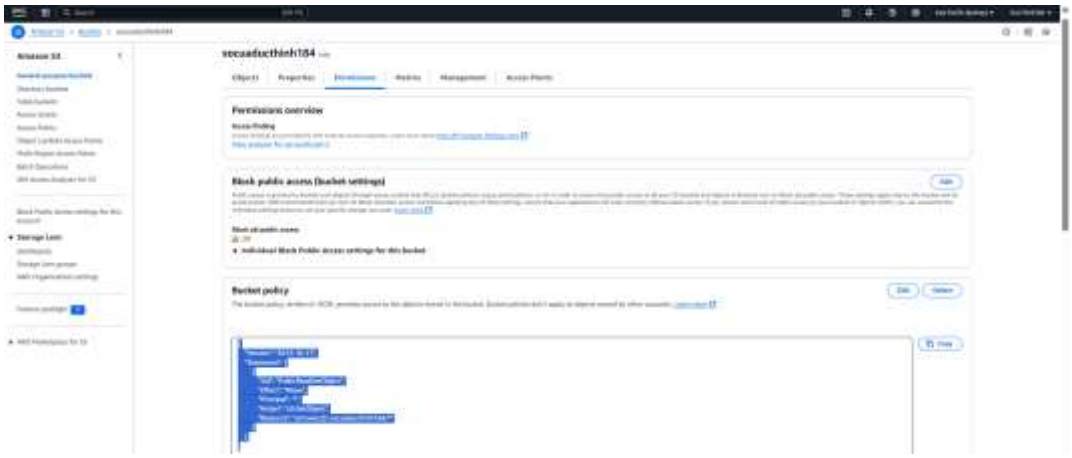
Trong dịch vụ S3, tạo 1 Bucket để chứa dữ liệu. Trong Bucket đó, tạo thêm 2 objects là **results** để lưu trữ ảnh kết quả nhận diện và **uploads** để lưu trữ ảnh người dùng uploads lên.



Hình 3. 19 Giao diện S3 trên AWS

Tiếp theo trong giao diện Bucket của S3, chọn **Permissions**, vào khung Bucket policy và viết đoạn Json này để ảnh kết quả và ảnh lưu trữ không hiển thì bằng Json mà hiển thị bằng ảnh.

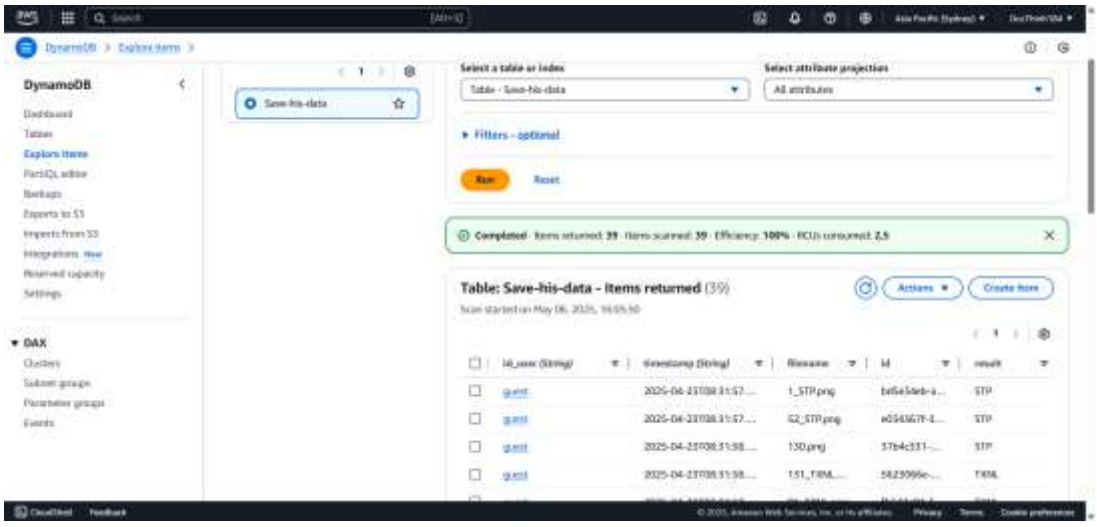
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::xocuaducthinh184/*"
    }
  ]
}
```



Hình 3. 20 Nơi viết chương trình để hiển thị ảnh trên S3

❖ Chọn DynamoDB và thiết lập để làm nơi lưu trữ lịch sử ảnh kết quả nhận diện của người dùng.

Trong dịch vụ DynamoDB, tạo 1 Table, sau khi tạo Table, Explore items sẽ là nơi lưu trữ lịch sử của người dùng.

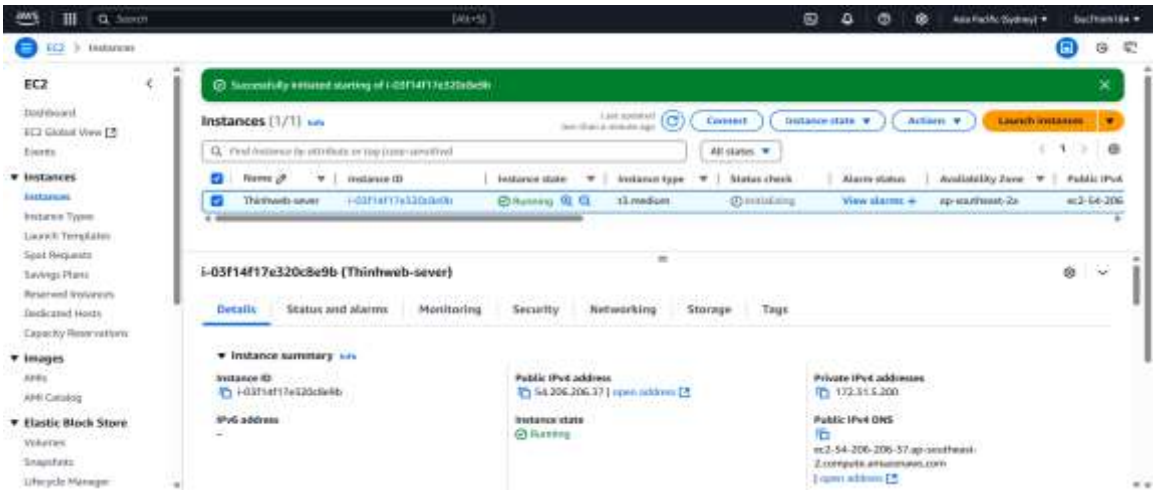


Hình 3. 21 Giao diện DynamoDB trên AWS

❖ Chọn EC2 và thiết lập để làm bộ não xử lý ảnh nhận diện trả kết quả về cho người dùng.

- Trong dịch vụ EC2, tạo Instant :
  - Trong **Amazon Machine Image**, lựa chọn môi trường Ubuntu để cài đặt môi trường cho AI.
  - Ở mục **Instance type**, chọn t3.medium (4Gb Ram) để đáp ứng được dung lượng xử lý của YOLO.

- Tạo **key pair** và lưu về máy để kết nối máy ảo trên EC2 với máy chủ.
- Trong mục **Network settings**, chọn hết 3 mục Allow SSH traffic from, Allow HTTPS traffic from the internet, Allow HTTP traffic from the internet để public ảnh trả về người dùng và kết nối với máy chủ ảo.
- Mục **Configure storage**, chọn dung lượng bộ nhớ Rom là 25Gb để chứa thoải mái các thử viện và model cần dùng.

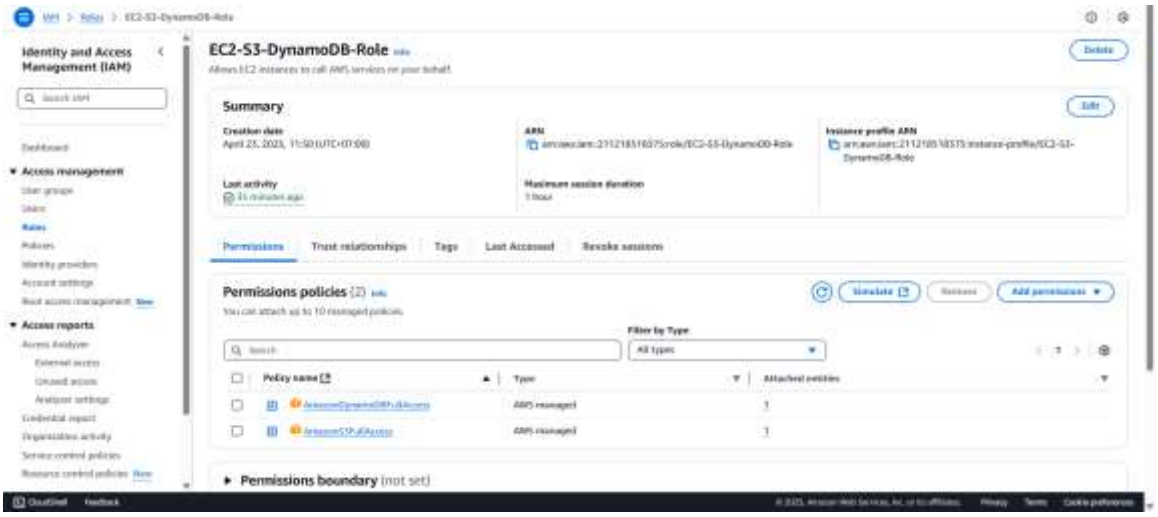


Hình 3. 22 Giao diện EC2 trên AWS

- Sau khi hiển thị giao diện Instances trong EC2 đã tạo, vào **Edit inboud rules** trong **Security groups** và thêm 1 cổng port 5000 để chạy Flask trên EC2.

❖ Chọn IAM để cấp quyền truy của S3 và DynamoDB cho EC2.

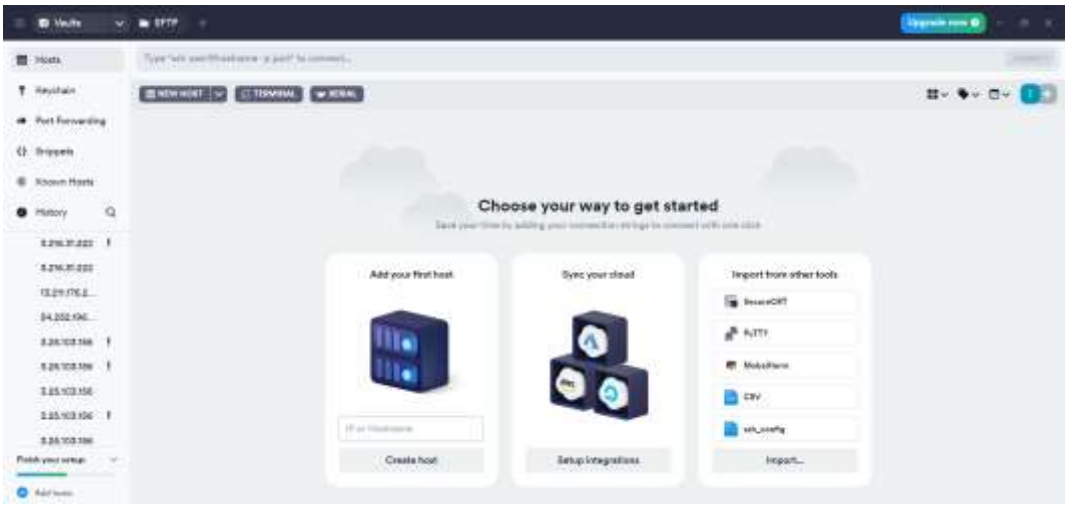
- Vào mục **Users** để tạo 1 user cấp quyền, chọn AmazonS3FullAccess và AmazonDynamoDBFullAccess.



Hình 3. 23 Giao diện IAM trên AWS

b. Tiến hành kết nối máy chủ với máy ảo trên EC2 bằng SSH với Termius :

Giao diện Termius :



Hình 3. 24 Giao diện termius

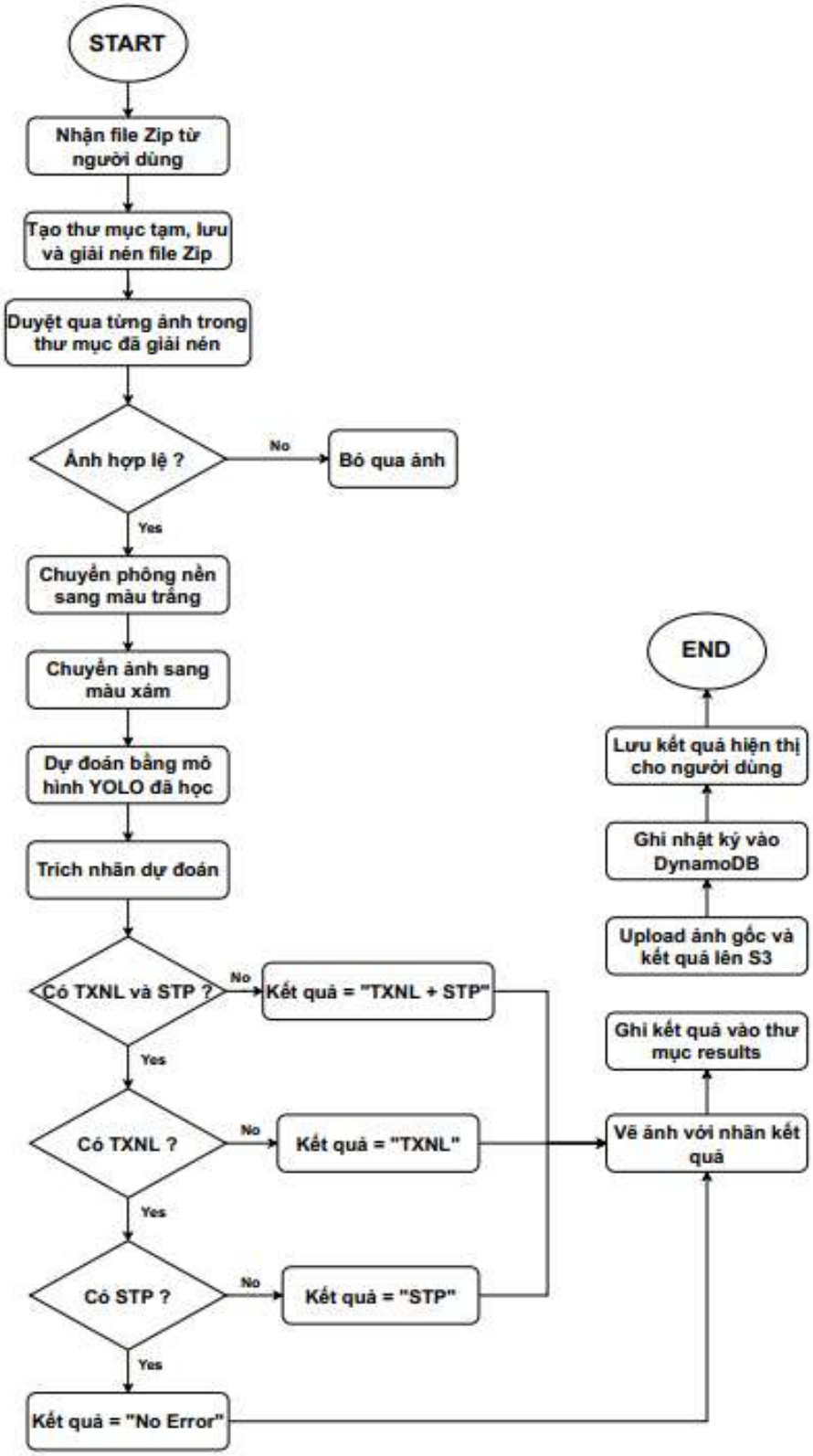
❖ Cấu hình địa chỉ, user và keypair để kết nối với EC2.

- Create host để tạo máy chủ ảo
  - Mục **Address** sẽ là địa chỉ Public IPv4 address trên Instant của EC2.
  - **Username** là ubuntu (môi trường đã tạo trên EC2).
  - Sau khi Connect file Key pair đã tải về khi tạo Instant trên EC2 và chọn Continue.
  - Giao diện kết nối thành công với máy chủ EC2.

Hình 3. 25 Giao diện điều khiển máy chủ ảo trên EC2 của Termius

c. Viết chương trình Flask websever để người dùng gửi ảnh và giao diện web :

Lưu đồ thuật toán phần xử lý :



Hình 3. 26 Lưu đồ thuật toán phần xử lý

Lưu đồ thuật toán gửi kết quả cho người dùng :

Hình 3. 27 Lưu đồ thuật toán trả kết quả cho người dùng

Chương trình :

(Dưới đây chỉ trích ra để giải thích những đoạn chương trình chính, toàn bộ chương trình sẽ được đưa vào phần phụ lục).

❖ Import thư viện :

```
from flask import Flask, render_template, request, send_file
import os, zipfile, shutil, uuid
from werkzeug.utils import secure_filename
from PIL import Image
import numpy as np
import cv2
from ultralytics import YOLO
import boto3
from datetime import datetime
from rembg import remove
```

- Giải thích :
  - Flask: Dùng để tạo web server.
  - Os, zipfile, shutil, uuid: Quản lý file, thư mục, giải nén ZIP, tạo mã UUID.
  - Secure_filename: Giúp đặt tên file an toàn.
  - PIL (Image): Xử lý ảnh.

- Numpy, cv2: Chuyển ảnh giữa định dạng PIL ↔ OpenCV.
- Ultralytics.YOLO: Dùng để load và chạy mô hình YOLOv8.
- Boto3: Tương tác với AWS (S3 và DynamoDB).
- Datetime: Ghi log thời gian.
- Rembg : Xoá phông

❖ Khởi tạo Flask và AWS :

```
app = Flask(__name__)
S3_BUCKET = 'xocuaducthinh184'
DYNAMO_TABLE = 'Save-his-data'
s3 = boto3.client('s3', region_name='ap-southeast-2')
dynamodb = boto3.resource('dynamodb', region_name='ap-southeast-2')
table = dynamodb.Table(DYNAMO_TABLE)
```

- Giải thích :
  - Tạo link web.
  - Khai báo Bucket S3 và bảng DynamoDB.
  - Tạo client AWS để upload file.

❖ Load model YOLO :

```
model = YOLO("Model_AI.pt")
```

- Giải thích : Tải mô hình AI đã học để dự đoán lỗi.

❖ Hàm upload file lên S3 :

```
def upload_to_s3(file_path, s3_path):
    s3.upload_file(file_path, S3_BUCKET, s3_path)
```

- Giải thích : Dùng boto3 để upload file lên Bucket S3 theo đường dẫn **s3_path.**

❖ Hàm ghi nhật ký vào DynamoDB :

```
def log_to_dynamodb(filename, result_text, id_user="guest"):
…
```

Giải thích : Ghi kết quả nhận diện của mỗi ảnh vào DynamoDB với ID người dùng, tên file, lỗi phát hiện và thời gian.

❖ Giao diện web để người dùng gửi ảnh :

```
@app.route("/")
def index():
```

```
return render_template("index.html", images=[])
```

- Giải thích : Giao diện web được viết bằng ngôn ngữ HTML để người dùng gửi ảnh lên nhận diện và tải ảnh nhận diện về nếu muốn.

❖ Nhận file zip, xử lý và nhận lỗi :

```
@app.route("/upload", methods=["POST"])
def upload_zip():
```

- Giải thích : Trong đoạn chương trình trên sẽ có phần chính như sau :
  - Tạo thư mục tạm và giải nén file ZIP.
  - Tạo thư mục lưu kết quả.
  - Xử lý từng ảnh.
  - Xoá nền sang phông trắng và chuyển về ảnh xám
  - Lọc kết quả qua NMS.
  - Lấy nhãn phát hiện và xác định lỗi
  - Vẽ kết quả và lưu ảnh.
  - Upload lên S3 và ghi lại nhật kí hoạt động.
  - Tạo link xem kết quả

❖ Tạo nơi tải ZIP kết quả :

```
@app.route("/download_zip")
def download_zip():
```

- Giải thích : Tạo file ZIP chứa ảnh đã nhận diện lỗi và gửi về cho người dùng.

❖ Chạy Flask :

```
if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

- Giải thích : Chạy flask tên EC2, cho phép người dùng truy cập từ bên ngoài.

d. Tiến hành deploy model và chương trình websever tạo giao diện web cho người dùng gửi ảnh :

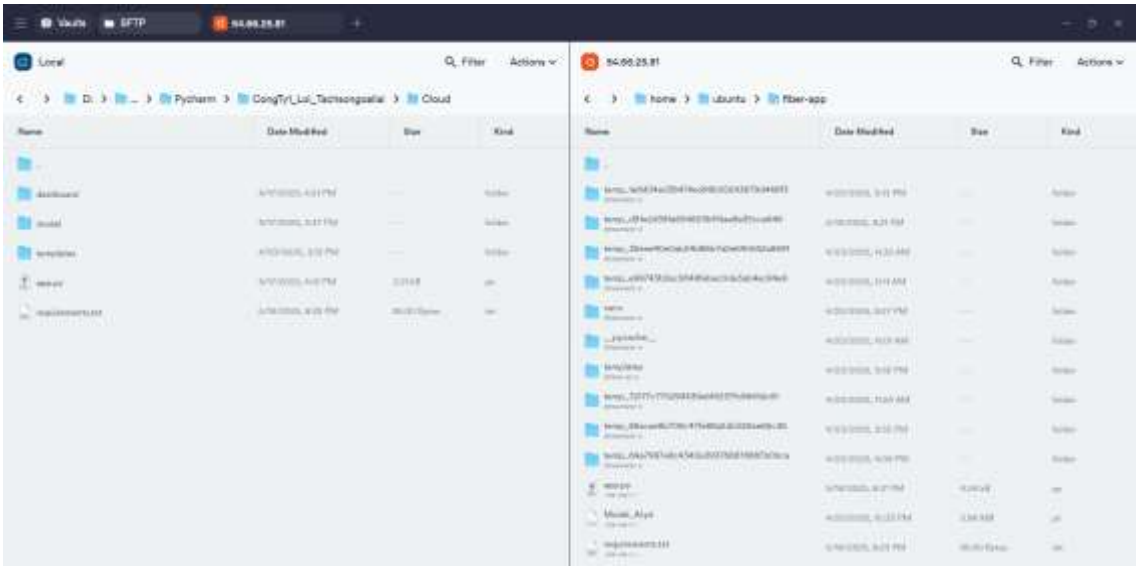
❖ Tạo thư mục và môi trường ảo :

```
mkdir fiber-app && cd fiber-app
python3 -m venv venv
source venv/bin/activate
```

❖ Tạo file requirements.txt chứa các thư viện để chạy chương trình trên máy ảo :

- Các thư viện : Flask, opencv, ultralytics, torch, numpy, pillow, gunicorn, rembg, onnxruntime.

❖ Tiến hành deploy model, chương trình và, file thư viện từ máy chủ lên EC2:



Hình 3. 28 Giao diện giữa máy chủ Local và máy chủ ảo của EC2

3.4. Chi phí đầu tư cho đề tài :

3.4.1. Huấn luyện mô hình :

- Dùng GG Colab phiên bản miễn phí.

3.4.2. Cloud :

- EC2 : chọn phiên bản máy ảo t3.medium kích thước 4GB Ram cần dùng để xử lý ảnh với YOLOv8n.
- Chi phí : ~ 0.06$ / giờ (Khoảng 1.120.000 đồng / tháng).

3.5. Kết luận chương 3 :

Ở chương 3, nhóm em đã thực hiện tính toán và phân tích khả năng nhận diện, huấn luyện mô hình. Đi sâu vào thiết kế phần mềm của mô hình, lập trình cho mô hình có khả năng học tăng dần. Thiết kế websever, cloud và tạo các dịch vụ cần thiết để người dùng truy cập, nhận diện và lưu lịch sử.

Chương 4 nhóm sẽ tiến mô phỏng và đánh giá thực nghiệm mô hình.

CHƯƠNG 4 : KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

4.1. Kết quả và đánh giá quá trình huấn luyện :

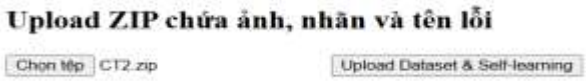
4.1.1. Tiến hành huấn luyện và kết quả đạt được sau huấn luyện :

a. Quá trình huấn luyện :

Tiến hành chạy flask trên Google colab đưa ra websever upload Zip chứa dataset để huấn luyện mô hình.

Hình 4. 1 Chạy chương trình đưa ra link websever

Giao diện websever hiển thị nơi upload Zip.

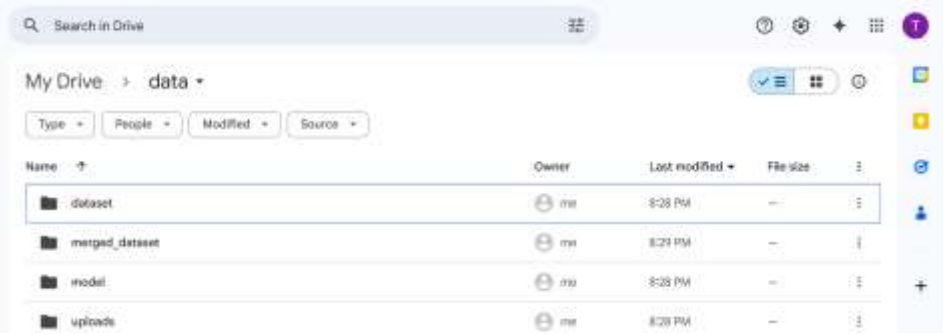


Hình 4. 2 Giao diện upload dữ liệu huấn luyện

Sau khi nhấn tải Zip lên và nhấn upload mô hình sẽ lưu bộ dataset về, tạo các thư mục chứa bộ dữ liệu, model sau khi học xong và tiến hành học.

Hình 4. 3 Mô hình đang học dữ liệu mới

Các bộ dữ liệu và model sẽ được tạo và lưu vào Google Drive.



Hình 4. 4 Giao diện Google Drive lưu trữ dữ liệu

Bộ dữ liệu đưa vào sẽ được đặt tên theo thời gian mà người dùng upload lên cho mô hình học

Theo *hình 4.5* thì mô hình được upload với thời gian : Ngày 1 tháng 6 năm 2025, lúc 13h 29 phút 13 giây (Theo múi giờ UTC+0).

Hình 4. 5 Dữ liệu được lưu trữ theo thời gian thực

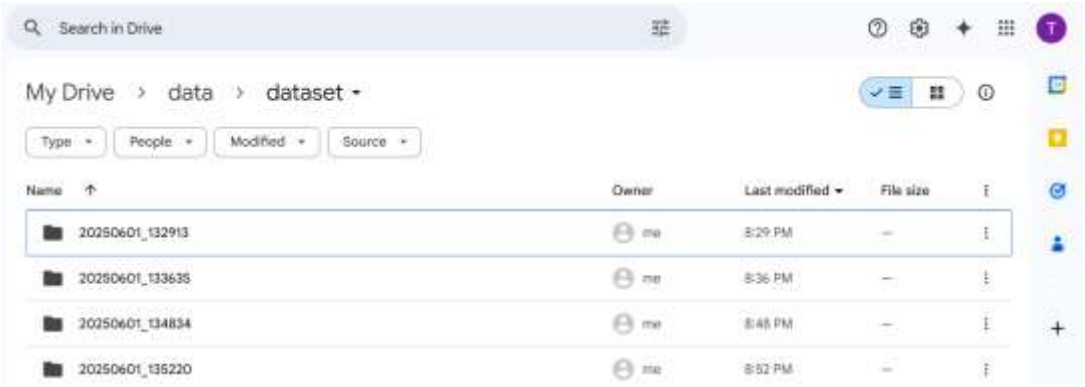
Sau khi mô hình học xong giao diện websever sẽ hiển thị : "Đã hoàn thành huấn luyện".

Hình 4. 6 Websever thông báo mô hình huấn luyện xong

Mô hình sau khi học xong sẽ được lưu vào thư mục model của Google Drive.

Hình 4. 7 Model sau huấn luyện được lưu trên Google Drive

Tiếp tục upload cho mô hình học hết 4 bộ dữ liệu như đã chuẩn bị ở **Chương 2**.



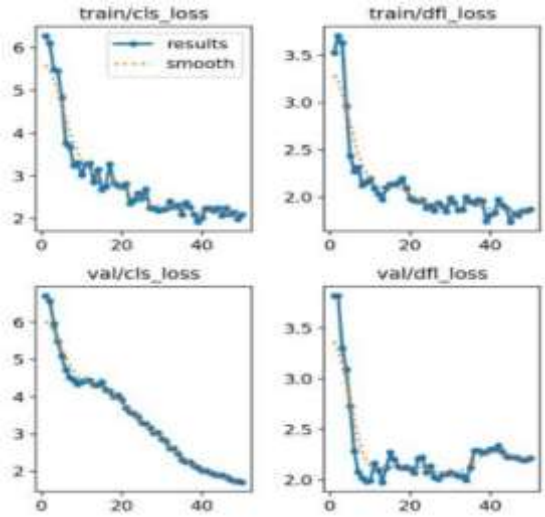
Hình 4. 8 Tất cả bộ dữ liệu huấn luyện được lưu trữ trên Google Drive

b. Kết quả đạt được sau huấn luyện :

❖ Model 1 (chỉ có dữ liệu lỗi TXNL với những sợi chỉ nhỏ) :

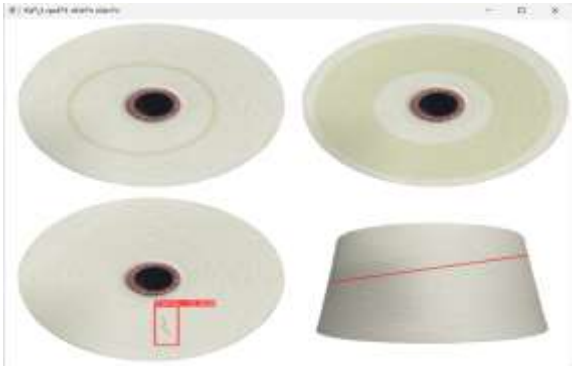
Đồ thị *train/cls_loss* và *val/cls_loss* : đường đồ thị giảm từ 0 đến 20 epoch cho thấy mô hình đang học tốt, khả năng phân loại cải thiện theo thời gian.

Đồ thị *train/dfl_loss* và *val/dfl_loss* : đường đồ thị giảm nhanh từ 0 đến 15 epoch, sau đó giao động nhẹ ở mức thấp.



Hình 4. 9 Đồ thị các hàm mất mát của model 1

Sau khi huấn luyện mô hình, model đưa ra chỉ nhận diện được được lỗi TXNL với sợi chỉ nhỏ màu đỏ, còn lại các lỗi khác trên cuộn sợi thì mô hình không nhận diện được và không khoanh vùng nào khác trên cuộn sợi.

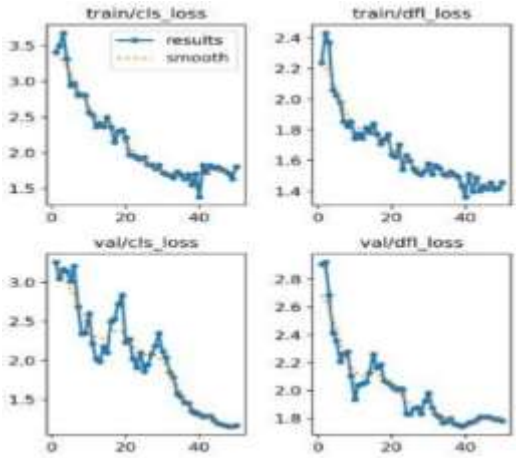


Hình 4. 10 Kết quả nhận diện của model 1

- ❖ Model 2 (học được lỗi TXNL với những sợi chỉ nhỏ và lỗi STP bị lẫn với một chút thành phần khác) :

Đồ thị *train/cls_loss* và *val/cls_loss* : Đồ thị train/cls_loss giảm rõ rệt và đều ở 45 epoch đầu cho thấy mô hình học khá ổn định trong việc phân loại. Đồ thị val/cls_loss dao động khá nhiều trong khoảng 0 đến 25 epoch cho thấy ở giai đoạn đầu mô hình chưa ổn định phân loại dữ liệu không huấn luyện nhưng về giai đoạn sau thì giảm dần ổn định.

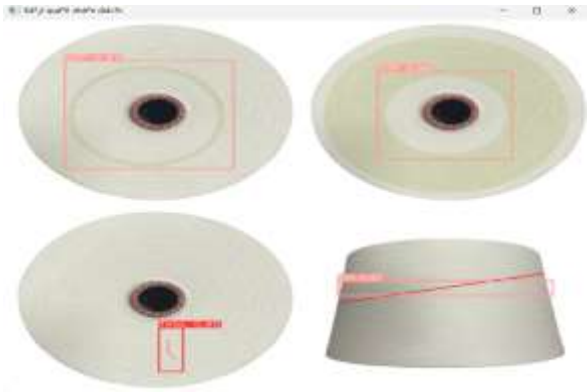
Đồ thị *train/dfl_loss* và *val/dfl_loss* : Đồ thị train/cls_loss giảm đều có vai dao động nhỏ cho thấy mô hình huấn luyện bouding box tốt. Đồ thị val/dfl_loss giảm mạnh nhưng có dao động mạnh hơn so với train/cls_loss, về sau giao động giảm.



Hình 4. 11 Đồ thị các hàm mất mát của model 2

Mô hình sau khi học thêm tệp Zip chứa dữ liệu lỗi STP bị lẫn với một chút thành phần khác thì đã nhận diện được cả lỗi TXNL và STP và khoanh vùng lỗi chính xác nhưng ở mức độ các lỗi nhỏ. Tại lỗi TXNL sợi chỉ lớn thì mô hình nhận diện nhầm là

STP, còn lỗi STP bị sai nhiều thành phần thì mô hình khoanh vùng không đúng vào trọng tâm lỗi.

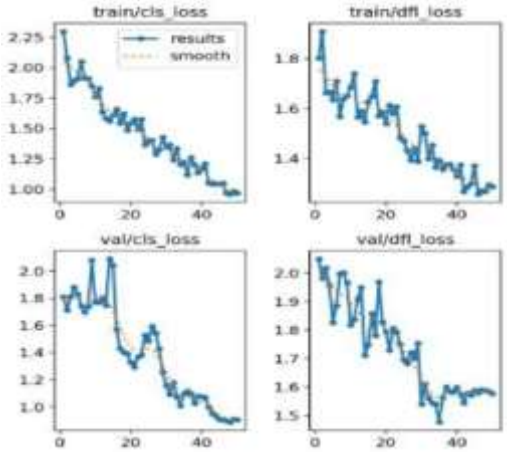


Hình 4. 12 Kết quả nhận diện của model 2

- ❖ Model 3 (học được các lỗi TXNL dài ngắn ở các mức độ khác nhau, lỗi STP bị lẫn một chút thành phần khác) :

Đồ thị *train/cls_loss* và *val/cls_loss* : Đồ thị train/cls_loss giảm đều trong cả 50 epoch, giao động ít cho thấy mô hình huấn luyện tốt. Đồ thị val/cls_loss có vài đỉnh nhọn nhưng cũng giảm dần ở epoch 20 trở đi.

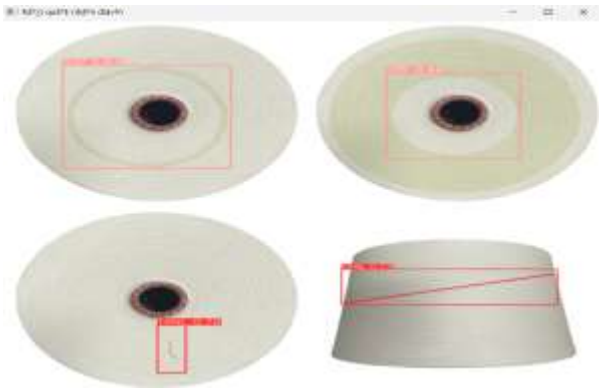
Đồ thị *train/dfl_loss* và *val/dfl_loss* : Đồ thị train/dfl_loss có dao động nhẹ nhưng vẫn giảm rõ ràng cho thấy mô hình huấn luyện tốt dự đoán vị trí. Đồ thị val/ dfl_loss giảm từ 2.0 xuống 1.5 nhưng có giao động mạnh, tuy vậy xu hướng vẫn giảm nên chấp nhận được.



Hình 4. 13 Đồ thị các hàm mất mát của model 3

Trong lần này, model đưa ra đã học được tốt khi nhận diện được lỗi TXNL ở các kích thước sợi bị lẫn vào khác nhau, khoanh vùng đúng toàn bộ sợi lẫn dài và ngắn. Lỗi STP

thì còn chưa nhận diện được lỗi trộn nhiều thành phần và khoanh vùng không đúng trọng tâm.

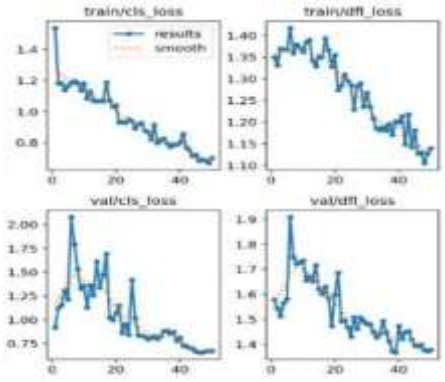


Hình 4. 14 Kết quả nhận diện của model 3

- ❖ Model 4 (học được các lỗi TXNL ở các mức độ khác nhau, lỗi STP bị lẫn cả ít và nhiều thành phần) :

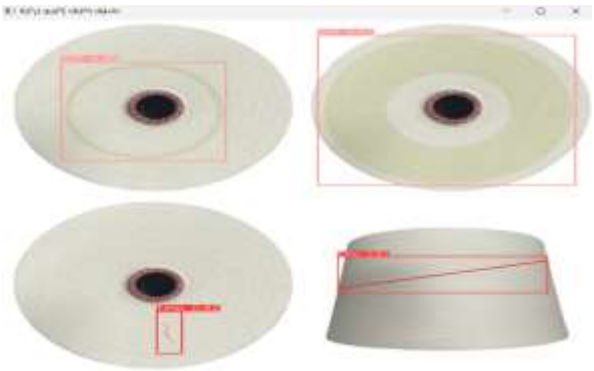
Đồ thị *train/cls_loss* và *val/cls_loss* : Đồ thị train/cls_loss giảm đều, có giao động nhẹ nhưng không đáng kể cho thấy mô hình huấn luyện tốt. Đồ thị val/cls_loss giao động mạnh hơn có lúc lên đến 2.0 nhưng sau cũng giảm dần cho thấy mô hình vẫn phân loại được các dữ liệu không được huấn luyện.

Đồ thị *train/dfl_loss* và *val/dfl_loss* : Đồ thị train/dfl_loss có tăng nhẹ nhưng cũng giảm đều trong 50 epoch cho thấy mô hình có cải thiện vị trí dự đoán Bouding box. Đồ thị val/dfl_loss có giao động mạnh ở khoảng 10 epoch đầu nhưng cũng có xu hướng giảm.



Hình 4. 15 Đồ thị các hàm mất mát của model 4

Sau khi cho mô hình huấn luyện đầy đủ các lỗi, model đưa ra đã nhận diện và đưa ra chính xác lỗi TXNL và lỗi STP. Các Bouding box cũng khoanh vùng chính xác tại điểm lỗi.



Hình 4. 16 Kết quả nhận diện của model 4

4.1.2. Đánh giá mô hình huấn luyện :

❖ Ưu điểm :

- Hiệu quả nhận diện của mô hình sau mỗi giai đoạn huấn luyện với các dữ liệu đưa vào từng đợt rất tốt, nhận diện được các lỗi từ đơn giản đến phức tạp.
- Mô hình không bị quên kiến thức cũ khi học dữ liệu mới, kết quả test thấy được mô hình phân biệt rõ ràng các loại lỗi, kể cả khi 2 lỗi xuất hiện cùng lúc.
- Triển khai trên Google Colab linh hoạt, huấn luyện mô hình nhanh và tiết kiệm chi phí với bản Colab miễn phí.

❖ Hạn chế :

- Nhóm em khảo sát thực tế và thu thập dữ liệu để đưa ra bộ dữ liệu lỗi huấn luyện mô hình trong tầm hiểu biết nên bộ dữ liệu vẫn chưa được đa dạng.
- Mô hình nhận diện tốt được với những dữ liệu đưa vào học, còn dữ liệu có cùng lỗi nhưng có hình dạng khác đặc biệt thì mô hình vẫn còn nhận diên chưa đủ tốt.
- Phụ thuộc nhiều vào chất lượng và độ đa dạng của tập dữ liệu huấn luyện.

4.2. Kết quả và đánh giá mô hình nhận diện cho người dùng trên Cloud :

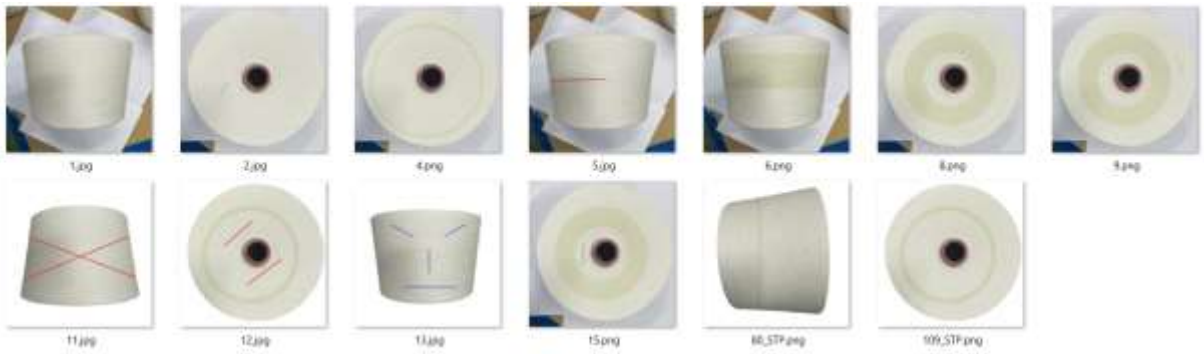
4.2.1. Tiến hành public link và kết quả nhận diện ảnh trả về cho người dùng :

Chạy chương trình đưa ra link websever trên máy chủ ảo http://ec2-52-65-159-226.ap-southeast-2.compute.amazonaws.com:5000/

Hình 4. 17 Chạy link public cho người dùng upload ảnh nhận diện

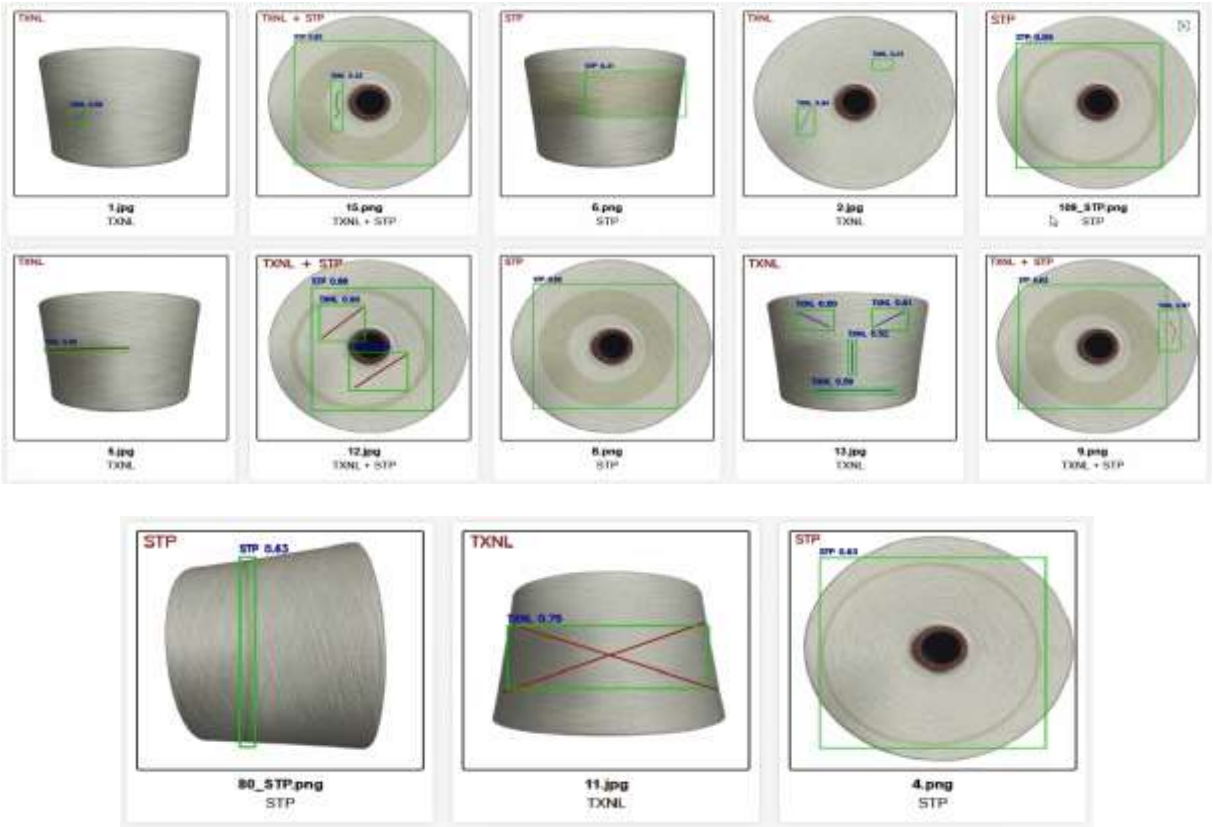
Ảnh ban đầu của người dùng khi chưa nhận diện.

Nén những ảnh lỗi chưa được nhận diện vào file “test.zip” để upload.



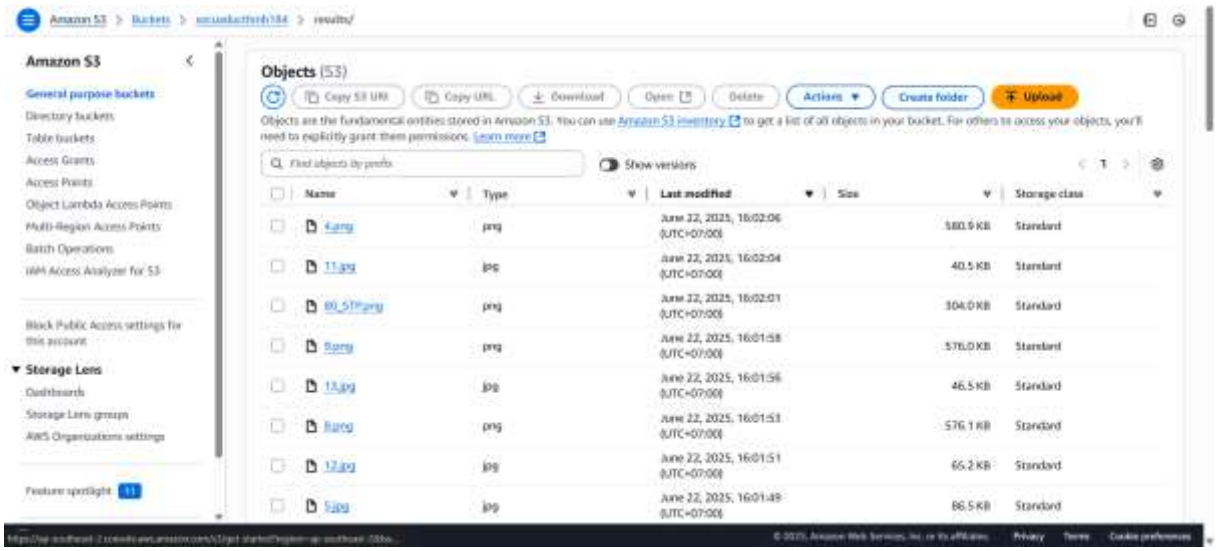
Hình 4. 18 Bộ ảnh chưa được nhận diện

Giao diện upload file Zip và kết quả ảnh nhận diện khi người dùng tải ảnh lên và cho phép người dùng lưu kết quả.



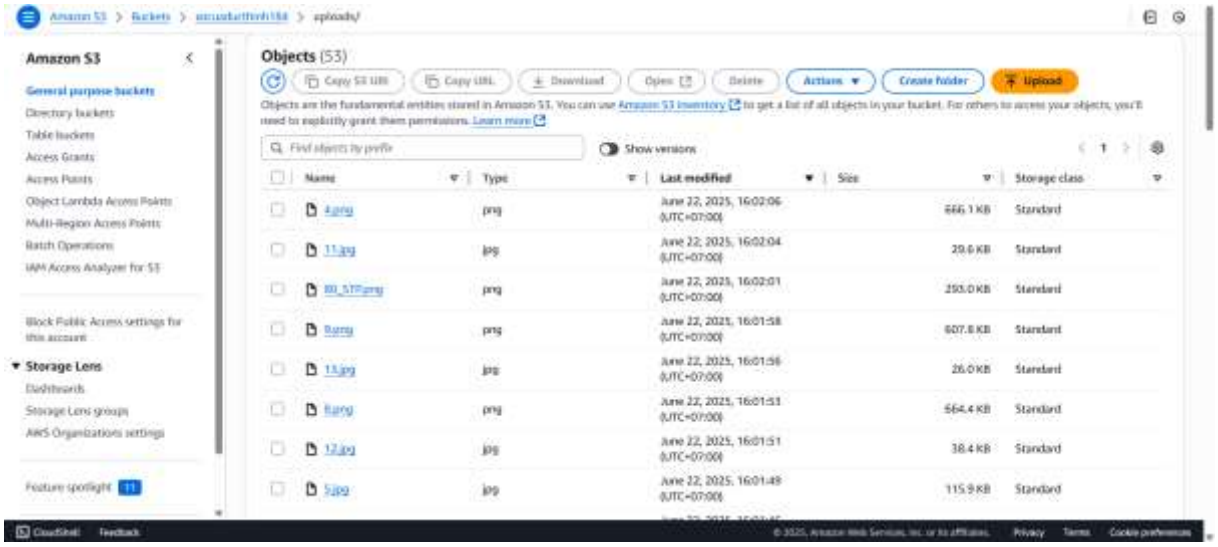
Hình 4. 19 Bộ ảnh của người dùng sau khi nhận diện

Kết quả ảnh nhận diện được lưu vào S3 trong file results.



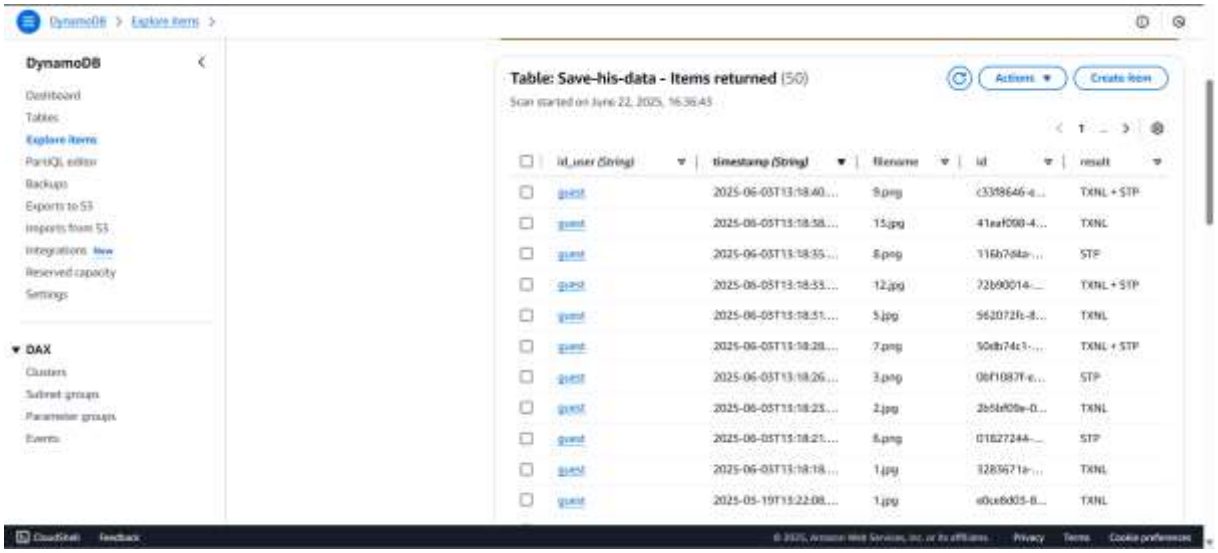
Hình 4. 20 Kết quả ảnh trên S3

Ảnh upload từ người dùng được lưu vào S3 trong file uploads.



Hình 4. 21 Ảnh upload từ người dùng trên S3

Lịch sử nhận diện và thông tin người dùng được lưu vào DynamoDB.



Hình 4. 22 Thông tin nhận người dùng và lịch sử nhận diện trên DynamoDB

4.2.2. Đánh giá kết quả nhận diện trên Cloud :

❖ Ưu điểm :

- Hệ thống hoạt động ổn định trên EC2 24/7, xử lý và trả kết quả nhanh.
- Ảnh gốc upload và ảnh kết quả được lưu trên S3 theo cấu trúc riêng biệt (results và uploads).
- Giao diện thân thiện, giúp người dùng lưu kết quả tải về dưới dạng ZIP.

❖ Nhược điểm :

- Hiệu suất phụ thuộc vào EC2, mô hình nhận diện càng nặng thì chi phí càng cao, khối lượng ảnh nhận diện nhiều có thể trả kết quả chậm.

4.3. Kết luận chương 4 :

Chương 4 đã tiến hành đánh giá hiệu quả của mô hình qua các giai đoạn huấn luyện và triển khai trên môi trường cloud. Kết quả thử nghiệm cho thấy:

- Mô hình học tăng dần giúp cải thiện đáng kể khả năng nhận diện lỗi từ đơn giản đến phức tạp.
- Hệ thống triển khai trên AWS hoạt động ổn định, cho phép người dùng gửi ảnh từ xa, nhận kết quả nhanh và lưu trữ thông tin lịch sử rõ ràng.
- Việc xử lý ảnh và lưu trữ kết quả thông qua S3 và DynamoDB tạo nền tảng tốt cho việc mở rộng hệ thống và phân tích dữ liệu lỗi theo thời gian thực.

KẾT LUẬN

Trong bối cảnh ngành công nghiệp dệt may Việt Nam ngày càng phát triển và hướng tới tự động hóa, việc ứng dụng trí tuệ nhân tạo vào kiểm tra chất lượng côn sợi là một hướng đi cần thiết và mang tính thực tiễn. Đề tài "Mô hình đánh giá chỉ tiêu chất lượng sản xuất sợi" đã bước đầu giải quyết được bài toán phát hiện lỗi côn sợi một cách tự động, nhanh chóng và hiệu quả.

Thông qua việc áp dụng mô hình học sâu YOLOv8, nhóm đã xây dựng thành công hệ thống nhận diện lỗi côn sợi với khả năng phát hiện các lỗi thường gặp như tách xơ ngoại lai và sai thành phần. Hệ thống được tích hợp với giao diện web trực quan, cho phép người dùng tải ảnh và nhận kết quả xử lý ngay trên nền tảng điện toán đám mây AWS. Ngoài ra, các dịch vụ như EC2, S3 và DynamoDB đã được triển khai để đảm bảo khả năng xử lý, lưu trữ và quản lý dữ liệu hiệu quả.

Kết quả thực nghiệm cho thấy hệ thống có khả năng vận hành ổn định, phát hiện lỗi với độ chính xác cao trên tập dữ liệu thực tế, đáp ứng tốt mục tiêu đề ra ban đầu.

Đóng góp của đề tài :

- Thiết kế và triển khai thành công một hệ thống nhận diện lỗi côn sợi tích hợp AI và cloud.
- Ứng dụng mô hình YOLOv8 tối ưu cho bài toán thị giác máy tính trong công nghiệp dệt.
- Đề xuất quy trình huấn luyện và triển khai mô hình trên nền tảng AWS một cách hiệu quả và tiết kiệm chi phí.

Đề xuất hướng phát triển :

- Trong tương lai, hệ thống có thể được mở rộng để nhận diện nhiều loại lỗi khác nhau bằng cách huấn luyện thêm với các tập dữ liệu đa dạng hơn.
- Tăng cường khả năng học liên tục (continual learning) để hệ thống có thể cập nhật lỗi mới từ dữ liệu thực tế.

TÀI LIỆU THAM KHẢO

[1] Alibaba Cloud, "Trí tuệ nhân tạo của Alibaba đạt độ chính xác hơn 90% trong kiểm tra chất lượng vải," Medium, 2020 [trực tuyến], có tại: https://alibaba-cloud.medium.com/alibaba-ai-achieves-identification-accuracy-of-more-than-90-for-textile-quality-inspection-8262b15025f4. [truy cập ngày 25 tháng 4 năm 2025].

[2] SOLOMON 3D, "Ứng dụng AI trong kiểm tra sợi – Giải pháp SolVision," SOLOMON 3D, 2025 [trực tuyến], có tại: https://www.solomon-3d.com/case-studies/solvision/yarn-inspection-ai. [truy cập ngày 26 tháng 4 năm 2025].

[3] Nguyễn Tiến Đạt, "Giới thiệu xử lý ảnh," Viblo, 2019 [trực tuyến], có tại: https://viblo.asia/p/tuan-1-gioi-thieu-xu-ly-anh-yMnKMdEQ57P. [truy cập ngày 8 tháng 5 năm 2025].

[4] VNPT AI, "Computer Vision là gì? Ứng dụng thị giác máy tính trong thực tiễn," VNPT AI, 2025 [trực tuyến], có tại: https://vnptai.io/vi/blog/detail/computer-vision-la-gi. [truy cập ngày 8 tháng 5 năm 2025].

[5] Chung Pham Van., "Deep Learning – Tìm hiểu về mạng tích chập CNN," Viblo, 2020 [trực tuyến], có tại: https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2. [truy cập ngày 12 tháng 5 năm 2025].

[6] Abin Timilsina, "YOLOv8 Architecture Explained," Medium, 2024 [trực tuyến], có tại: https://abintimilsina.medium.com/yolov8-architecture-explained-a5e90a560ce5. [truy cập ngày 10 tháng 5 năm 2025].

[7] Đỗ Gia Huy, Phạm Diệu Nguyên, "Giới thiệu YOLOv8 và các bước huấn luyện mô hình nhận diện vật thể," HackMD, 2023 [trực tuyến], có tại: https://hackmd.io/@58ZC49ZfS86wYX--LRGGOg/Viewperm. [truy cập ngày 24 tháng 4 năm 2025].

[8] Ultralytics, "Thư viện Ultralytics – YOLOv8," GitHub, truy cập trực tuyến tại: https://github.com/ultralytics/ultralytics. [truy cập ngày 17 tháng 5 năm 2025].