**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**======\*\*\*======**



**BÁO CÁO THỰC NGHIỆM**

**HỌC PHẦN XỬ LÝ ẢNH SỐ VÀ THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI**

**XÂY DỰNG HỆ THỐNG NHẬN DIỆN BỆNH TRÊN LÁ CÂY (NẤM, VÀNG LÁ, SÂU BỆNH,…)**

|  |  |
| --- | --- |
| **GVHD:**  **Lớp:**  **Nhóm:**  **Sinh viên:** | GV. Lương Thị Hồng Lan  20242IT6072002  03  Cao Thành Lâm - 2022602901  Dương Văn Lộc - 2022606870  Nguyễn Phương Nam - 2022603105  Trần Văn Nhã - 2022603089 |

***Hà Nội – 2025***

**MỤC LỤC**

[CHƯƠNG I: KIẾN THỨC CƠ SỞ 1](#_Toc201534009)

* 1. [Giới thiệu học máy 1](#_Toc201534010)
     1. [Khái niệm 1](#_Toc201534011)

[1.1.2. Các loại học 2](#_Toc201534012)

* 1. [Các loại bài toán trong xử lý ảnh 3](#_Toc201534013)
     1. [Phân lớp ảnh 3](#_Toc201534014)
     2. [Nhận dạng đối tượng 4](#_Toc201534015)
     3. [Phân đoạn ảnh 5](#_Toc201534016)
     4. [Phát hiện bất thường 6](#_Toc201534017)
  2. [Ngôn ngữ lập trình và thư viện sử dụng 10](#_Toc201534018)
  3. [Các thuật toán phổ biến 11](#_Toc201534019)

[1.4.1. CNN 11](#_Toc201534020)

[1.4.2. KNN 13](#_Toc201534021)

[CHƯƠNG II: MÔ HÌNH 18](#_Toc201534022)

[2.1. Phát biểu bài toán 18](#_Toc201534023)

[2.1.1. Mục tiêu tổng quát 18](#_Toc201534024)

[2.1.2. Các bài toán con 18](#_Toc201534025)

[2.1.3. Giới hạn của bài toán 19](#_Toc201534026)

[2.2. Mô hình huấn luyện 20](#_Toc201534027)

[2.2.1. Quy trình xử lý ảnh đầu vào 20](#_Toc201534028)

[2.2.2. Kiến trúc mô hình CNN 23](#_Toc201534029)

[2.2.3. Cách chọn tham số cho CNN 29](#_Toc201534030)

[2.2.4. Chi tiết huấn luyện 29](#_Toc201534031)

[2.4. Công cụ, môi trường phát triển 35](#_Toc201534032)

[2.4.1. Môi trường huấn luyện 35](#_Toc201534033)

[2.4.2. Giao diện người dùng 36](#_Toc201534034)

[2.4.3. Xử lý phía máy chủ 36](#_Toc201534035)

[2.5. Triển khai mô hình lên Website 37](#_Toc201534036)

[2.5.1. Kiến trúc hệ thống 37](#_Toc201534037)

[2.5.2. Quy trình hoạt động 38](#_Toc201534038)

[2.5.3. Giao diện người dùng 38](#_Toc201534039)

[2.5.4. Tổ chức dự án 39](#_Toc201534040)

[CHƯƠNG III: THỰC NGHIỆM 40](#_Toc201534041)

[3.1. Dữ liệu 40](#_Toc201534042)

[3.1.1. Nguồn dữ liệu 40](#_Toc201534043)

[3.1.2. Tiền xử lý 40](#_Toc201534044)

[3.1.3. Làm giàu dữ liệu 41](#_Toc201534045)

[3.1.4. Phân đoạn ảnh và đặc trưng 43](#_Toc201534046)

[3.2. Mô hình huấn luyện 45](#_Toc201534047)

[3.3. Kết quả thực nghiệm 45](#_Toc201534048)

[3.4. Đánh giá mô hình 47](#_Toc201534049)

[KẾT LUẬN 52](#_Toc201534050)

[TÀI LIỆU THAM KHẢO 55](#_Toc201534051)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1 Thuật toán CNN 11](#_Toc201533968)

[Hình 1.2 Thuật toán KNN 13](#_Toc201533969)

[Hình 1.3 Thuật toán cây quyết định 16](#_Toc201533970)

[Hình 2.1 Cấu trúc bài toán 19](#_Toc201533971)

[Hình 2.2 Kiến trúc mạng CNN 25](#_Toc201533972)

[Hình 2.3 Convolutional layer 26](#_Toc201533973)

[Hình 2.4 Fully connected layer 28](#_Toc201533974)

[Hình 2.5 Mô hình quy trình bài toán 29](#_Toc201533975)

[Hình 2.6 Kiến trúc mạng nơ-ron tích chập 30](#_Toc201533976)

[Hình 2.7 Sơ đồ kiến trúc của mô hình CNN sử dụng trong huấn luyện 31](#_Toc201533977)

[Hình 2.8 Quá trình huấn luyện 32](#_Toc201533978)

[Hình 2.9 Biểu đồ Accuracy theo từng epoch 33](#_Toc201533979)

[Hình 2.10 Biểu đồ Loss theo từng epoch 33](#_Toc201533980)

[Hình 2.11 Cấu trúc thư mục 34](#_Toc201533981)

[Hình 2.12 Quy trình xử lý phía máy chủ 37](#_Toc201533982)

[Hình 2.13 Giao diện của mô hình khi được triển khai 38](#_Toc201533983)

[Hình 2.14 Cấu trúc dự án 39](#_Toc201533984)

[Hình 3.1 ảnh sau khi xử lý 41](#_Toc201533985)

[Hình 3.2 Kết quả ảnh được làm giàu 42](#_Toc201533986)

[Hình 3.3 Ảnh sau khi xử lý và phân cụm MeanShift 44](#_Toc201533987)

[Hình 3.4 Biểu đồ Accuracy / Loss của Resnet50, MobileNetV2, CNN 45](#_Toc201533988)

[Hình 3.5 Biểu đồ so sánh Accuracy / Loss giữa Resnet50, MobileNetV2, CNN 46](#_Toc201533989)

[Hình 3.6 Quy trình huấn luyện mô hình MobileNetV2 47](#_Toc201533990)

[Hình 3.7 Kết quả huấn luyện mô hình MobileNetV2 48](#_Toc201533991)

[Hình 3.8 Quy trình huấn luyện mô hình Resnet50 49](#_Toc201533992)

[Hình 3.9 kết quả huấn luyện mô hình Resnet50 50](#_Toc201533993)

[Hình 3.10 Quy trình và kết quả huấn luyện mô hình nhận diện ảnh lá 51](#_Toc201533994)

**DANH MỤC BẢNG BIỂU**

[Bảng 1.1 Ưu điểm và hạn chế của thuật toán CNN 12](#_Toc201533995)

[Bảng 1.2 Ưu điểm và hạn chế của thuật toán KNN 14](#_Toc201533996)

[Bảng 1.3 Ưu điểm và hạn chế của thuật toán Cây quyết định 17](#_Toc201533997)

[Bảng 2.1 Công cụ thư viện sử dụng 35](#_Toc201533998)

[Bảng 2.2 Ngôn ngữ sử dụng cho giao diện 36](#_Toc201533999)

[Bảng 2.3 Một số thư viện sử dụng trong Backend 36](#_Toc201534000)

[Bảng 3.1 Các kĩ thuật làm giàu dữ liệu 41](#_Toc201534001)

[Bảng 3.2 Ưu, nhược điểm của các mô hình 46](#_Toc201534002)

# CHƯƠNG I: KIẾN THỨC CƠ SỞ

* 1. **Giới thiệu học máy**

### **1.1.1. Khái niệm**

Học máy (Machine Learning) là một lĩnh vực thuộc trí tuệ nhân tạo, tập trung vào việc nghiên cứu và phát triển các kỹ thuật cho phép hệ thống “học” tự động từ dữ liệu để giải quyết các vấn đề cụ thể.

Hiểu đơn giản, thuật ngữ này nói tới việc con người dạy máy tính nâng cao khả năng thực hiện các tác vụ cụ thể. Cụ thể là cung cấp các dữ liệu và thuật toán có sẵn để máy tính đưa ra dự đoán hoặc tự ra các quyết định. Thông thường, con người chỉ cần lập trình phần mềm với các dòng lệnh cụ thể để máy tính hiểu và thực hiện. Với Machine Learning, máy tính sẽ tự “học” cách giải quyết công việc thông qua những dữ liệu đã được thu thập và cung cấp.

Theo Tom Mitchell trong cuốn sách “Machine Learning” xuất bản năm 1997, Machine Learning như 1 chương trình, nhiệm vụ của nó là thực hiện 1 nhiệm vụ T nào đó, khi thực hiện xong, ta thu được trải nghiệm E. Nhờ vào việc học hỏi trải nghiệm E, ta có thể thay đổi (hoặc không) để tiến tới thực hiện task T+1, và nhằm cải thiện hiệu suất P.

Lấy ngay ví dụ là AlphaGo – 1 chương trình máy tính chuyên để chơi cờ vây. T chính là việc AlphaGo chơi cờ với các người chơi khác nhau, E chính là kinh nghiệm AlphaGo thu được sau khi chơi các ván cờ, còn P chính là xác suất AlphaGo thắng ván tiếp theo. AlphaGo sẽ liên tục chơi (thực hiện nhiệm vụ T) để cập nhật kinh nghiệm E và nâng cao xác suất thắng P.

### **1.1.2. Các loại học**

Máy tính có thể học tập theo nhiều phương pháp khác nhau, trong đó 3 cách phổ biến nhất là:

**Học máy có giám sát**

Học máy có giám sát (Supervised Machine Learning) là phương pháp mà trong đó máy tính được học từ dữ liệu đã được đánh dấu trước, để phát triển các thuật toán có khả năng phân loại hoặc dự đoán kết quả một cách chính xác. Kỹ thuật này thường được áp dụng cho các bài toán phân lớp (Classification).

Ví dụ: Chúng ta có một tập dữ liệu gồm các hình ảnh của chó và mèo. Trong giai đoạn đầu, máy tính được huấn luyện để nhận biết các hình ảnh này. Khi quá trình huấn luyện hoàn tất, máy tính có thể nhận diện và dự đoán chính xác loại động vật trong hình ảnh mới được cung cấp, dựa trên việc phân tích các đặc điểm như hình dạng và màu sắc. Quá trình này chính là nhận dạng đối tượng trong máy học có giám sát.

**Học máy không giám sát**

Trái ngược với học máy có giám sát, máy học không giám sát (Unsupervised Learning) sử dụng thuật toán để phân tích và phân cụm các dữ liệu không có nhãn. Phương pháp này tự động tìm kiếm các mô hình và cấu trúc ẩn trong dữ liệu mà không yêu cầu sự hỗ trợ từ con người.

Ví dụ: Máy học không giám sát nổi bật với khả năng nhận diện các điểm chung và khác biệt trong dữ liệu, làm cho nó trở thành công cụ hữu ích trong việc phân tích dữ liệu khám phá, phân loại khách hàng, phát triển chiến lược bán chéo (Cross-sell), nhận dạng hình ảnh, và nhiều ứng dụng khác.

**Học máy bán giám sát**

Trong bối cảnh dữ liệu phát triển nhanh chóng và không có cách nào để chúng được gắn nhãn kịp thời, đó là lý do học máy bán giám sát (Semi-supervised Learning) trở nên rất quan trọng. Phương pháp này kết hợp việc sử dụng dữ liệu đã được gắn nhãn và chưa gắn nhãn để huấn luyện máy tính.

Trong quá trình huấn luyện, con người sử dụng một tập dữ liệu có nhãn nhỏ hơn để chỉ dẫn máy tính cách phân loại và trích xuất từ một lượng lớn dữ liệu chưa được gắn nhãn. Semi-supervised Learning giúp giải quyết các vấn đề khi không có đủ dữ liệu gắn nhãn cho việc huấn luyện máy học có giám sát.

Ví dụ: Nó được ứng dụng rộng rãi trong các lĩnh vực như nhận diện giọng nói, phân loại nội dung web và tài liệu.

## **Các loại bài toán trong xử lý ảnh**

### **Phân lớp ảnh**

Khái niệm: Phân lớp ảnh liên quan đến việc sử dụng các thuật toán để phân chia hình ảnh thành các nhóm dựa trên các đặc điểm khác nhau. Thông thường, các thuật toán này sử dụng học máy hoặc học sâu để đạt được độ chính xác cao. Phân lớp ảnh có thể là phân lớp nhị phân, nơi một ảnh được gán cho một trong hai nhãn, hoặc phân lớp đa lớp với nhiều nhãn hơn.

Các Phương Pháp Phân Lớp Ảnh:

* Phân Lớp Thủ Công

Đây là phương pháp truyền thống, trong đó chuyên gia con người phân tích và phân lớp hình ảnh dựa trên kinh nghiệm và kiến thức cá nhân. Phương pháp này thường tốn nhiều thời gian và không đạt được độ chính xác cao khi xử lý một lượng lớn dữ liệu.

* Phân Lớp Tự Động sử dụng Máy Học

Các mô hình học máy như SVM (Support Vector Machines), k-NN (k-Nearest Neighbors), và Random Forest được sử dụng rộng rãi. Những mô hình này yêu cầu một lượng dữ liệu huấn luyện đại diện để hoạt động hiệu quả.

* Phân Lớp Dựa Trên Mạng Nơ-ron Sâu

Với sự phát triển của trí tuệ nhân tạo, mạng nơ-ron sâu đã trở thành phương pháp phổ biến cho phân lớp ảnh. Các kiến trúc như CNN (Convolutional Neural Networks) có khả năng tự động trích xuất các đặc điểm của hình ảnh và phân loại chúng với độ chính xác cao.

### **Nhận dạng đối tượng**

Khái niệm: Nhận dạng đối tượng (Object Recognition) là một lĩnh vực thuộc trí tuệ nhân tạo và thị giác máy tính, cho phép máy tính xác định và phân loại các đối tượng trong hình ảnh hoặc video. Công nghệ này ngày càng phổ biến trong các ứng dụng như xe tự lái, giám sát an ninh, y tế và thương mại điện tử.

Các phương pháp nhận dạng đối tượng:

1. Học sâu (Deep Learning)
   * Sử dụng mạng nơ-ron tích chập (CNN) để tự động học các đặc trưng từ hình ảnh.
   * Các mô hình phổ biến: YOLO (You Only Look Once), Faster R-CNN, SSD (Single Shot MultiBox Detector).
2. Nhận dạng dựa trên đặc trưng (Feature-Based Recognition)
   * Sử dụng thuật toán như SIFT (Scale-Invariant Feature Transform), ORB (Oriented FAST and Rotated BRIEF) để phát hiện và ghép nối các đặc trưng trong ảnh.
   * Hiệu quả với hình ảnh chứa đối tượng có độ biến đổi nhỏ về hình dạng và kích thước.
3. Phương pháp dựa vào thị giác máy tính truyền thống
   * Dựa vào màu sắc, kết cấu hoặc hình dạng để phân loại đối tượng.
   * Áp dụng trong các hệ thống đơn giản như phát hiện biển báo giao thông.
4. Nhận dạng bằng mô hình lai (Hybrid Models)
   * Kết hợp nhiều phương pháp để đạt độ chính xác cao hơn.
   * Ví dụ: Kết hợp CNN với mạng hồi quy (RNN) để phân tích đối tượng trong video.

### **Phân đoạn ảnh**

Khái niệm: Phân đoạn hình ảnh (Image Segmentation) là một kỹ thuật cơ bản trong xử lý hình ảnh kỹ thuật số và thị giác máy tính có thể phân chia một hình ảnh số thành nhiều phân đoạn (vùng hoặc đối tượng) để đơn giản hóa và phân tích hình ảnh bằng cách tách nó thành các thành phần có ý nghĩa.

Thông qua việc này, quá trình xử lý hình ảnh được thực hiện một cách hiệu quả hơn bằng cách tập trung vào các vùng cụ thể cần quan tâm. Một nhiệm vụ phân đoạn hình ảnh điển hình trải qua các bước sau:

* Nhóm các pixel trong hình ảnh dựa trên những đặc điểm chung như màu sắc, cường độ hoặc kết cấu.
* Gán nhãn cho mỗi pixel, chỉ ra sự thuộc về của nó vào một phân đoạn hoặc đối tượng cụ thể.

Kết quả là một hình ảnh đã được phân đoạn, thường được trực quan hóa dưới dạng mặt nạ hoặc lớp phủ làm nổi bật các phân đoạn khác nhau.

Một số phương pháp phân đoạn hình ảnh phổ biến:

1. Phân đoạn dựa trên ngưỡng (Thresholding):  
    Là phương pháp đơn giản nhất, chia ảnh thành các vùng dựa trên giá trị cường độ của pixel. Phổ biến nhất là phương pháp Otsu Thresholding.
2. Phân đoạn theo vùng (Region-based Segmentation):  
    Bao gồm các thuật toán như Region Growing và Region Splitting & Merging, trong đó các vùng được mở rộng hoặc chia nhỏ dựa trên sự tương đồng giữa các pixel lân cận.
3. Phân đoạn dựa trên ranh giới (Edge-based Segmentation):  
    Sử dụng các thuật toán phát hiện biên như Canny, Sobel, hay Laplacian of Gaussian để xác định các đường viền phân cách giữa các vùng.
4. Phân cụm (Clustering):  
    Sử dụng các thuật toán như K-means, Mean Shift hoặc Gaussian Mixture Models (GMMs) để phân nhóm pixel dựa trên đặc trưng không gian hoặc màu sắc.
5. Mô hình Markov ngẫu nhiên (MRF) và CRF:  
    Áp dụng các mô hình xác suất để đưa ra phân đoạn tối ưu, có khả năng duy trì thông tin không gian giữa các pixel.
6. Phân đoạn bằng học sâu (Deep Learning-based Segmentation):  
    Các mô hình như FCN (Fully Convolutional Networks), U-Net, SegNet, và DeepLab đang trở thành xu hướng chính trong các ứng dụng hiện đại. Chúng cho phép phân đoạn chính xác đến từng pixel trong các tác vụ phức tạp như y tế, tự lái, nông nghiệp thông minh,…

### **Phát hiện bất thường**

Khái niệm: Phát hiện bất thường hay còn gọi là Anomaly Detection là quá trình phân tích dữ liệu để xác định các điểm bất thường trong tập dữ liệu, không tuân theo mẫu dữ liệu tiêu chuẩn. Việc áp dụng phát hiện bất thường có thể thấy trong nhiều lĩnh vực, bao gồm phát hiện gian lận, lỗi hệ thống, xâm nhập trái phép, giám sát tình trạng thiết bị và nhận diện sự kiện trong mạng cảm biến. Mặc dù những bất thường có thể xảy ra thường xuyên, nhưng đôi khi chúng lại tiềm ẩn rủi ro nghiêm trọng, chẳng hạn như các cuộc tấn công mạng hoặc hành vi gian lận.

Dữ liệu bất thường là những điểm dữ liệu hoặc sự kiện có đặc điểm khác biệt rõ rệt so với mô hình chuẩn. Khi một giá trị nằm ngoài phạm vi của tập dữ liệu thông thường, nó có thể là dấu hiệu ban đầu của sự cố hệ thống, vi phạm an ninh hoặc lỗ hổng bảo mật mới xuất hiện. Bất thường trong dữ liệu có thể bao gồm các giá trị không nhất quán, dữ liệu trùng lặp, lỗi chèn dữ liệu, dữ liệu tải lên không đầy đủ hoặc bị xóa một cách đột ngột trong cơ sở dữ liệu.

Các phương pháp Anomaly Detection phổ biến:

Tùy vào loại dữ liệu và mục tiêu phân tích, doanh nghiệp có thể lựa chọn nhiều phương pháp Anomaly Detection khác nhau để phát hiện bất thường, từ các phương pháp trực quan hóa đơn giản đến các thuật toán máy học tiên tiến. Mỗi phương pháp đều có ưu điểm riêng trong việc xác định và xử lý các điểm dữ liệu bất thường, giúp nâng cao độ chính xác và hiệu quả trong phân tích. Hãy cùng VNPT AI tìm hiểu chi tiết những phương pháp phát hiện bất thường phổ biến nhất hiện nay:

**1. Trực quan hóa dữ liệu (Visualization)**

Trực quan hóa dữ liệu là một công cụ mạnh mẽ giúp các nhà khoa học dữ liệu nhanh chóng nhận diện các điểm bất thường. Bằng cách vẽ biểu đồ hoặc đồ thị, họ có thể dễ dàng phát hiện các mẫu dữ liệu khác lạ hoặc xu hướng bất thường.

**2. Kiểm định thống kê (Statistical Tests)**

Khi tìm hiểu về anomaly detection là gì, chắc chắn không thể bỏ qua phương pháp Statistical Tests. Các kiểm định thống kê giúp phát hiện bất thường bằng cách so sánh dữ liệu quan sát được có khác biệt nhiều so với phân phối hoặc mô hình dự kiến. Các phương pháp được sử dụng phổ biến nhất là:

* Grubbs test: Xác định các điểm dữ liệu nằm ngoài phạm vi trung bình và độ lệch chuẩn.
* Kolmogorov-Smirnov test: Kiểm tra xem một tập dữ liệu có tuân theo một phân phối cụ thể hay không, chẳng hạn như phân phối chuẩn (normal distribution).

**3. Thuật toán máy học (Machine Learning Anomaly Detection)**

Thuật toán học máy được sử dụng để giúp phát hiện bất thường bằng cách học mô hình dữ liệu bình thường và xác định các điểm sai lệch. Một số thuật toán phổ biến bao gồm:

* Decision trees: Một dạng cây quyết định, Isolation Forest, là một phương pháp học tập tổ hợp (ensemble learning) giúp cô lập điểm bất thường bằng cách chọn ngẫu nhiên một đặc trưng và chia nhỏ dữ liệu nhiều lần.
* One-Class Support Vector Machine (SVM): One-Class SVM là một thuật toán phân loại được huấn luyện chỉ trên các điểm dữ liệu "bình thường", nhằm tạo ra một ranh giới bao quanh dữ liệu bình thường. Các điểm dữ liệu nằm ngoài ranh giới này được coi là bất thường.
* k-Nearest Neighbors (k-NN): k-NN là một thuật toán đơn giản phân loại một điểm dữ liệu dựa trên nhóm của k lân cận gần nhất. Nếu một điểm dữ liệu có quá ít điểm lân cận cùng nhóm, nó có thể bị coi là bất thường.
* Naive Bayesian: Phương pháp này dự đoán khả năng xảy ra của một sự kiện dựa trên các yếu tố liên quan.
* Autoencoders: Đây là một loại mạng nơ-ron nhân tạo sử dụng dữ liệu có dấu thời gian để dự đoán các mẫu dữ liệu và phát hiện các điểm bất thường không phù hợp với dữ liệu lịch sử.
* Local Outlier Factor (LOF): LOF là một thuật toán dựa trên mật độ, đo lường sự lệch mật độ cục bộ của một điểm dữ liệu so với các điểm lân cận. Các điểm có mật độ thấp hơn đáng kể so với hàng xóm của chúng được coi là bất thường.
* k-means clustering: k-means là một kỹ thuật dữ liệu thành các cụm, điểm nào không thuộc nhóm rõ ràng có thể là bất thường.

**Ứng dụng phát hiện bất thường (Anomaly Detection) trong bài toán nhận diện bệnh lá cây**

Trong bối cảnh nông nghiệp thông minh, bài toán nhận diện bệnh lá cây đóng vai trò quan trọng trong việc hỗ trợ người nông dân phát hiện sớm các dấu hiệu bệnh, từ đó đưa ra biện pháp xử lý kịp thời nhằm bảo vệ năng suất và chất lượng cây trồng. Bên cạnh các kỹ thuật phổ biến như phân loại hình ảnh (image classification) hoặc phân đoạn hình ảnh (image segmentation), thì phát hiện bất thường (Anomaly Detection) cũng là một hướng tiếp cận hiệu quả, đặc biệt hữu ích trong việc xử lý các trường hợp chưa từng xuất hiện trong dữ liệu huấn luyện.

Phát hiện bất thường là quá trình xác định các điểm dữ liệu không tuân theo quy luật chung hoặc khác biệt đáng kể so với phần còn lại của tập dữ liệu. Trong bài toán nhận diện bệnh lá cây, phương pháp này có thể được ứng dụng ở nhiều khía cạnh:

* **Phát hiện sớm bệnh lạ hoặc chưa có trong dữ liệu huấn luyện**: Mô hình học máy có thể được huấn luyện với dữ liệu lá cây khỏe mạnh, từ đó nhận biết các hình ảnh có dấu hiệu lạ là bất thường. Đây là trường hợp điển hình áp dụng các thuật toán như *Autoencoder*, *One-Class SVM* hoặc *Isolation Forest*.
* **Tăng độ chính xác cho hệ thống phân loại**: Trước khi đưa vào mô hình phân loại bệnh cụ thể, hệ thống có thể sử dụng Anomaly Detection như một bước tiền xử lý nhằm loại bỏ các ảnh nhiễu, ảnh bị hỏng, hoặc các mẫu không rõ ràng.
* **Giám sát cây trồng thời gian thực**: Trong các hệ thống sử dụng camera hoặc máy bay không người lái (drone) để giám sát vườn cây, kỹ thuật phát hiện bất thường có thể giúp hệ thống nhanh chóng phát hiện các dấu hiệu khác thường như lá bị đổi màu, xuất hiện đốm lạ hoặc hình thái không bình thường.
* **Tăng khả năng tổng quát hóa mô hình**: Bằng cách phát hiện và cô lập các điểm bất thường, hệ thống có thể tránh bị ảnh hưởng bởi các mẫu nhiễu hoặc outlier, từ đó nâng cao hiệu quả huấn luyện và độ tin cậy của mô hình.

Ví dụ: một Autoencoder được huấn luyện với các ảnh lá khỏe mạnh sẽ có khả năng tái tạo tốt những ảnh tương tự. Khi gặp ảnh có biểu hiện bất thường, sai số tái tạo sẽ tăng cao, cho phép mô hình xác định đây là ảnh nghi ngờ bị bệnh. Tương tự, các thuật toán như *k-means clustering* hoặc *LOF (Local Outlier Factor)* có thể được sử dụng để xác định các mẫu không phù hợp với nhóm dữ liệu đã biết.

## **Ngôn ngữ lập trình và thư viện sử dụng**

**Mô hình AI:** Python là ngôn ngữ phổ biến. Dùng TensorFlow/Keras hoặc PyTorch để xây dựng và huấn luyện CNN. Các thư viện hỗ trợ: OpenCV (xử lý ảnh), Albumentations (augmentation), scikit-learn (tiền xử lý, metric), PyTorch/TensorFlow Datasets (quản lý dữ liệu). Để triển khai trên thiết bị, có thể chuyển model sang định dạng ONNX hoặc TensorFlow Lite/CoreML.

**Backend/API:** Sử dụng Python (FastAPI, Flask) hoặc Node.js (Express) để xây dựng API phục vụ mô hình. FastAPI được ưa chuộng vì hiệu năng cao và tạo tài liệu tự động. Có thể dùng Flask với Uvicorn/Gunicorn để chạy WSGI. Các lựa chọn khác: Java (Spring Boot), Golang. Đóng gói API trong Docker để dễ triển khai.

**Frontend đa nền tảng:**

**Web:** React.js, Angular hoặc Vue.js để làm giao diện web. Có thể kết hợp PWA (Progressive Web App) để chạy tốt trên cả PC và điện thoại.

**Mobile:** Flutter (Dart) hoặc React Native (JavaScript/TypeScript) cho ứng dụng Android/iOS đa nền. Flutter hiện rất phổ biến vì một mã nguồn chạy tốt trên cả hai nền tảng.

**Desktop:** Có thể dùng Electron (JavaScript) hoặc sử dụng Flutter Desktop / .NET MAUI để tạo ứng dụng Windows/macOS/Linux. Nếu dùng Electron/React, có thể chia sẻ code với web.

**Hạ tầng vận hành:** Sử dụng Docker để container hóa API và model, dễ dàng triển khai trên cloud hoặc máy chủ cục bộ. Quản lý luồng công việc với Git, CI/CD (GitHub Actions, GitLab CI) để tự động build và deploy.

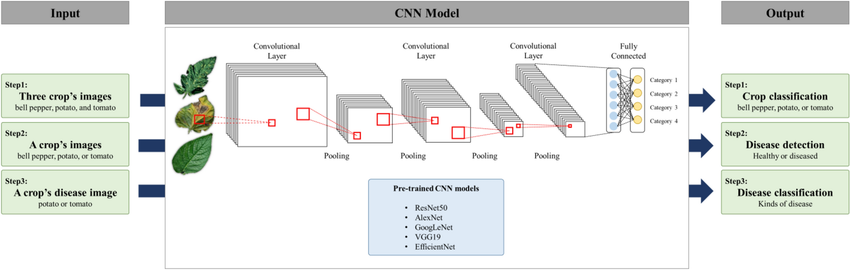
## **Các thuật toán phổ biến**

### **CNN**

**CNN** là một loại mạng nơ-ron sâu được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc dạng lưới, như hình ảnh. Trong bối cảnh nhận diện bệnh cây trồng, CNN hoạt động như sau:

1. **Thu thập dữ liệu**: Thu thập hình ảnh lá cây bị bệnh và khỏe mạnh.
2. **Tiền xử lý và trích xuất đặc trưng**: Sử dụng các lớp tích chập (convolutional layers) để tự động trích xuất các đặc trưng quan trọng từ hình ảnh, như màu sắc, kết cấu, hình dạng.
3. **Phân loại**: Các đặc trưng được trích xuất sẽ được đưa qua các lớp kết nối đầy đủ (fully connected layers) và lớp softmax để phân loại hình ảnh vào các lớp bệnh cụ thể.

CNN có khả năng học các đặc trưng phức tạp và trừu tượng từ dữ liệu hình ảnh, giúp cải thiện độ chính xác trong việc nhận diện bệnh cây trồng.



Hình 1.1 Thuật toán CNN

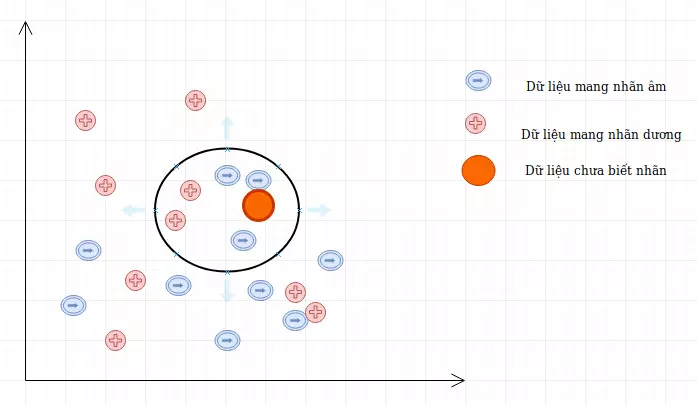
Bảng 1.1 Ưu điểm và hạn chế của thuật toán CNN

|  |  |
| --- | --- |
| **Ưu điểm** | **Hạn chế** |
| * **Tự động trích xuất đặc trưng**: Không cần thiết kế thủ công các đặc trưng, CNN có thể học trực tiếp từ dữ liệu hình ảnh. * **Độ chính xác cao**: CNN đã chứng minh hiệu quả vượt trội trong các nhiệm vụ phân loại hình ảnh, bao gồm nhận diện bệnh cây trồng. * **Khả năng tổng quát hóa tốt**: CNN có thể học các đặc trưng phân biệt giữa các lớp bệnh khác nhau, giúp mô hình hoạt động tốt trên dữ liệu mới. * **Khả năng xử lý dữ liệu lớn**: CNN có thể xử lý và học từ các tập dữ liệu hình ảnh lớn, cải thiện hiệu suất mô hình. | * **Yêu cầu dữ liệu lớn**: CNN cần một lượng lớn dữ liệu huấn luyện để đạt hiệu suất tốt, điều này có thể là thách thức trong lĩnh vực nông nghiệp. * **Chi phí tính toán cao**: Huấn luyện CNN đòi hỏi tài nguyên tính toán lớn, như GPU, và thời gian huấn luyện dài. * **Khó giải thích**: CNN thường được coi là "hộp đen", khó giải thích cách mô hình đưa ra quyết định, điều này có thể là vấn đề trong các ứng dụng yêu cầu tính minh bạch cao. * **Dễ bị overfitting**: Nếu không được điều chỉnh đúng cách, CNN có thể học quá mức từ dữ liệu huấn luyện và không tổng quát hóa tốt cho dữ liệu mới. |

### **1.4.2. KNN**

Thuật toán K-Nearest Neighbors (KNN) là một phương pháp học có giám sát (supervised learning) đơn giản nhưng hiệu quả, đặc biệt trong các bài toán phân loại. Trong bối cảnh nhận diện bệnh cây trồng, KNN hoạt động như sau:

1. **Thu thập dữ liệu:** Thu thập hình ảnh lá cây bị bệnh và khỏe mạnh.
2. **Tiền xử lý và trích xuất đặc trưng:** Sử dụng các kỹ thuật xử lý ảnh để trích xuất các đặc trưng như màu sắc, kết cấu, hình dạng từ hình ảnh lá cây.
3. **Tính toán khoảng cách**: Đối với một hình ảnh lá cây mới cần phân loại, thuật toán tính toán khoảng cách (thường là khoảng cách Euclidean) giữa đặc trưng của hình ảnh này với tất cả các mẫu trong tập huấn luyện.
4. **Tìm k láng giềng gần nhất:** Chọn ra k mẫu trong tập huấn luyện có khoảng cách gần nhất với mẫu mới.
5. **Phân loại:** Mẫu mới được gán nhãn theo đa số nhãn của k láng giềng gần nhất.



Hình 1.2 Thuật toán KNN

Bảng 1.2 Ưu điểm và hạn chế của thuật toán KNN

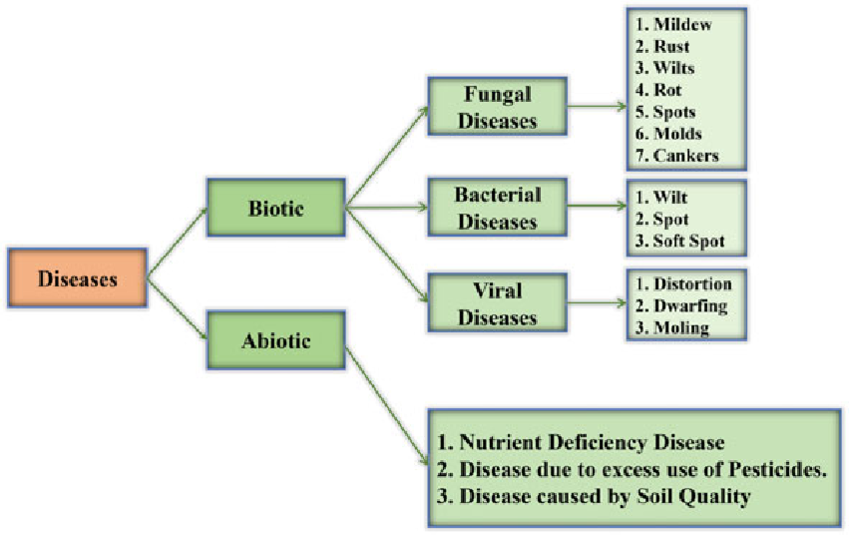
|  |  |
| --- | --- |
| **Ưu điểm** | **Hạn chế** |
| * **Đơn giản và dễ triển khai**: KNN là một trong những thuật toán dễ hiểu và dễ triển khai nhất trong học máy. * **Không cần huấn luyện mô hình**: KNN là một thuật toán dựa trên ví dụ (instance-based), không yêu cầu giai đoạn huấn luyện mô hình phức tạp. * **Hiệu quả với dữ liệu có cấu trúc rõ ràng**: KNN hoạt động tốt khi dữ liệu có cấu trúc và các lớp phân biệt rõ ràng. * **Linh hoạt với nhiều loại dữ liệu**: Có thể áp dụng cho cả dữ liệu số và phân loại, cũng như dữ liệu hình ảnh sau khi trích xuất đặc trưng. | * **Chi phí tính toán cao**: Với tập dữ liệu lớn, việc tính toán khoảng cách đến tất cả các mẫu trong tập huấn luyện có thể tốn nhiều thời gian và tài nguyên. * **Nhạy cảm với dữ liệu nhiễu**: Các mẫu nhiễu hoặc ngoại lai có thể ảnh hưởng đến kết quả phân loại. * **Hiệu suất giảm với dữ liệu có nhiều chiều**: Trong không gian có nhiều chiều (high-dimensional space), khoảng cách giữa các điểm trở nên kém phân biệt, ảnh hưởng đến hiệu quả của KNN. * **Cần lựa chọn k phù hợp**: Giá trị k không phù hợp có thể dẫn đến phân loại sai. |

***1.4.3. Cây quyết định***

Thuật toán **Cây quyết định** là một phương pháp học có giám sát (supervised learning) được sử dụng để phân loại hoặc hồi quy. Trong bối cảnh nhận diện bệnh cây trồng, thuật toán này hoạt động bằng cách:

1. **Thu thập dữ liệu**: Thu thập hình ảnh lá cây và các đặc trưng liên quan (màu sắc, hình dạng, kết cấu, v.v.).
2. **Tiền xử lý và trích xuất đặc trưng**: Sử dụng các kỹ thuật xử lý ảnh để trích xuất các đặc trưng quan trọng từ hình ảnh, chẳng hạn như màu sắc trung bình, mức độ xanh, hoặc các đặc trưng kết cấu như GLCM (Gray-Level Co-occurrence Matrix).
3. **Xây dựng cây quyết định**:
   * **Nút gốc (Root Node)**: Chọn đặc trưng phân chia tốt nhất dựa trên các chỉ số như Entropy hoặc Gini để bắt đầu xây dựng cây.
   * **Nút quyết định (Decision Nodes)**: Tại mỗi nút, dữ liệu được phân chia tiếp tục dựa trên các đặc trưng khác.
   * **Lá (Leaf Nodes)**: Mỗi lá đại diện cho một lớp đầu ra, chẳng hạn như loại bệnh cụ thể hoặc tình trạng khỏe mạnh.

Quá trình này tiếp tục cho đến khi đạt được một điều kiện dừng, chẳng hạn như khi tất cả các mẫu trong một nút thuộc cùng một lớp hoặc khi không còn đặc trưng nào để phân chia.



Hình 1.3 Thuật toán cây quyết định

Bảng 1.3 Ưu điểm và hạn chế của thuật toán Cây quyết định

|  |  |
| --- | --- |
| **Ưu điểm** | **Hạn chế** |
| * **Dễ hiểu và giải thích**: Mô hình cây quyết định có cấu trúc rõ ràng, dễ dàng diễn giải và giải thích cho người dùng, đặc biệt hữu ích trong lĩnh vực nông nghiệp. * **Xử lý cả dữ liệu số và phân loại**: Có khả năng xử lý cả dữ liệu liên tục (như mức độ xanh) và dữ liệu phân loại (như màu sắc lá). * **Không yêu cầu chuẩn hóa dữ liệu**: Không cần chuẩn hóa hoặc biến đổi dữ liệu đầu vào, giúp tiết kiệm thời gian tiền xử lý. * **Hiệu quả với tập dữ liệu nhỏ**: Có thể đạt được kết quả tốt ngay cả với tập dữ liệu huấn luyện nhỏ, miễn là dữ liệu đủ đại diện. * **Tự động chọn đặc trưng quan trọng**: Thuật toán tự động chọn ra các đặc trưng quan trọng nhất để phân chia dữ liệu. | * **Dễ bị overfitting**: Cây quá sâu hoặc quá phức tạp có thể dẫn đến việc mô hình học quá kỹ dữ liệu huấn luyện và không tổng quát hóa tốt cho dữ liệu mới. * **Không ổn định**: Thay đổi nhỏ trong dữ liệu huấn luyện có thể dẫn đến sự thay đổi lớn trong cấu trúc cây, ảnh hưởng đến độ tin cậy của mô hình. * **Thiên lệch với đặc trưng có nhiều giá trị**: Các đặc trưng có nhiều giá trị phân loại có thể được ưu tiên chọn làm nút phân chia, ngay cả khi chúng không thực sự quan trọng, dẫn đến mô hình thiên lệch. * **Hiệu suất thấp với dữ liệu lớn**: Với tập dữ liệu lớn và nhiều đặc trưng, cây quyết định có thể trở nên cồng kềnh và chậm chạp, ảnh hưởng đến hiệu suất. |

# CHƯƠNG II: MÔ HÌNH

## **2.1. Phát biểu bài toán**

### **2.1.1. Mục tiêu tổng quát**

Hệ thống được xây dựng nhằm nhận diện bệnh trên lá cây thông qua hình ảnh, phục vụ mục đích giám sát cây trồng, chẩn đoán sớm và hỗ trợ điều trị kịp thời trong nông nghiệp.

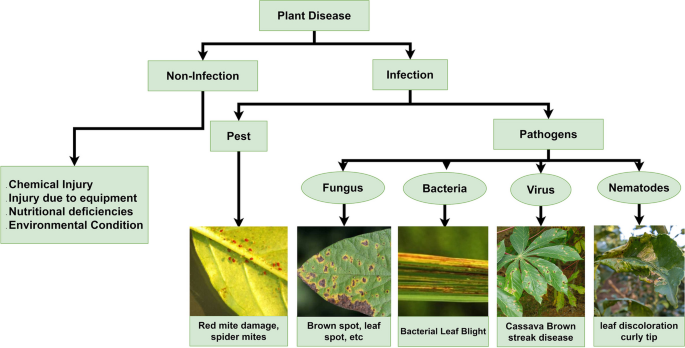
Cụ thể, với một hoặc nhiều ảnh đầu vào chứa hình ảnh lá cây, hệ thống cần:

* Dự đoán loại bệnh (nếu có)
* Ước lượng mức độ nghiêm trọng của tổn thương,
* Và trong trường hợp có ảnh từ nhiều thời điểm, phân tích tiến triển bệnh theo thời gian (lan rộng hay thu hẹp), đồng thời đưa ra gợi ý điều trị phù hợp.

### **2.1.2. Các bài toán con**

Bài toán phân loại (Image Classification)

* Đầu vào: Hình ảnh lá cây nghi ngờ có bệnh (ảnh đã qua bước tiền xử lý: cắt vùng lá, resize về kích thước cố định, chuẩn hóa).
* Đầu ra: Nhãn của loại bệnh, thuộc một trong 6 lớp sau: Bệnh đốm vi khuẩn, nấm, mốc sương, phấn trắng, đốm lá (Cháy lá), khỏe mạnh.
* Phương pháp: Mạng nơ-ron tích chập (CNN) để trích xuất đặc trưng từ ảnh và thực hiện phân loại.



Hình 2.1 Cấu trúc bài toán

Bài toán phát hiện vùng bệnh trên lá

### **2.1.3. Giới hạn của bài toán**

* Với đề tài trên, chúng em chỉ tập trung vào việc xử lý và phân tích ảnh lá cây với 5 loại bệnh phổ biến, bao gồm: Đốm vi khuẩn, nấm, mốc sương, phấn trắng, đốm lá (Cháy lá)



* Hệ thống không đảm bảo hiệu quả trong các trường hợp ảnh đầu vào có:
* Chất lượng thấp (nhòe, nhiễu mạnh, thiếu sáng)
* Đối tượng không rõ ràng (lá bị khuất, không đầy đủ, nền ảnh quá phức tạp)
* Không phải lá cây hoặc ảnh bị sai lệch dữ liệu

## **2.2. Mô hình huấn luyện**

### **2.2.1. Quy trình xử lý ảnh đầu vào**

*2.2.1.1. Các giai đoạn*

Quy trình xử lý ảnh đầu vào bao gồm các bước chính sau:

Thu nhận ảnh: Ảnh được thu thập từ các nguồn như máy ảnh kỹ thuật số, cảm biến, hoặc cơ sở dữ liệu. Định dạng ảnh đầu vào thường là RGB hoặc grayscale, tùy thuộc vào ứng dụng.

Tiền xử lý:

* Chuẩn hóa: Điều chỉnh kích thước và độ phân giải để đảm bảo tính thống nhất.
* Lọc nhiễu: Áp dụng bộ lọc Gaussian hoặc Median để loại bỏ nhiễu.
* Cân bằng sáng/tương phản: Sử dụng cân bằng histogram để cải thiện chất lượng ảnh.

Phân đoạn ảnh: Tách các vùng quan tâm (ROI) bằng các kỹ thuật như ngưỡng hoặc phân cụm, hỗ trợ việc phân tích các đối tượng cụ thể.

Trích xuất đặc trưng: Sử dụng các phương pháp như SIFT hoặc mạng nơ-ron tích chập (CNN) để xác định các đặc trưng quan trọng (cạnh, góc, kết cấu).

Xử lý nâng cao: Áp dụng các thuật toán như CNN hoặc Faster R-CNN để nhận diện đối tượng, hoặc các kỹ thuật như siêu phân giải để cải thiện chất lượng ảnh.

Hậu xử lý: Tinh chỉnh kết quả, làm mịn biên, và chuẩn bị đầu ra phù hợp với mục tiêu ứng dụng.

Đầu ra: Kết quả có thể là ảnh đã xử lý, nhãn phân loại, hoặc dữ liệu phân tích (tọa độ, đặc trưng).

*2.1.2.2. Các kĩ thuật biến đổi ảnh*

Trong lĩnh vực thị giác máy tính (Computer Vision), dữ liệu hình ảnh đầu vào đóng vai trò quyết định đến hiệu quả và độ chính xác của các mô hình học máy. Tuy nhiên, trong nhiều trường hợp, tập dữ liệu ban đầu không đủ lớn hoặc thiếu đa dạng, dẫn đến hiện tượng overfitting – khi mô hình học quá tốt trên dữ liệu huấn luyện nhưng kém hiệu quả trên dữ liệu thực tế. Để khắc phục vấn đề này, các kỹ thuật biến đổi ảnh (Image Augmentation/Transformation) được áp dụng nhằm sinh ra các phiên bản mới từ ảnh gốc mà vẫn giữ nguyên thông tin cơ bản, giúp mô hình có khả năng khái quát hóa tốt hơn.

Trong khuôn khổ đề tài, nhóm đã triển khai và sử dụng các kỹ thuật biến đổi ảnh cơ bản nhưng hiệu quả, bao gồm:

#### 1. Xoay ảnh (Rotation)

* Mục tiêu: Mô phỏng các góc nhìn khác nhau của vật thể nhằm giúp mô hình học được tính bất biến xoay (rotation invariance).
* Phương pháp: Ảnh đầu vào được xoay một góc ngẫu nhiên trong khoảng từ -30 đến +30 độ. Việc này đặc biệt hữu ích trong các bài toán nhận diện đối tượng hoặc chữ viết, nơi góc nhìn có thể thay đổi tùy vào bối cảnh thực tế.
* Tác động: Làm tăng khả năng nhận diện đối tượng từ các góc độ khác nhau.

2. Thay đổi độ sáng và độ tương phản (Brightness and Contrast Adjustment)

* Mục tiêu: Tăng khả năng chống chịu của mô hình với sự thay đổi ánh sáng trong điều kiện môi trường thực tế.
* Phương pháp: Điều chỉnh các giá trị pixel bằng cách áp dụng phép biến đổi tuyến tính hoặc gamma correction để làm ảnh sáng hơn hoặc tối hơn, đồng thời tăng/giảm độ tương phản.
* Tác động: Mô hình học được nhiều dạng ánh sáng khác nhau, từ đó hoạt động tốt trong điều kiện không đồng nhất về ánh sáng.

#### 3. Phóng to / thu nhỏ và cắt ảnh (Zoom & Random Crop)

* Mục tiêu: Mô phỏng việc vật thể ở các khoảng cách khác nhau hoặc không nằm chính giữa ảnh.
* Phương pháp: Cắt ảnh theo tỷ lệ ngẫu nhiên và sau đó resize về kích thước ban đầu, hoặc phóng to một vùng cụ thể trong ảnh.
* Tác động: Tăng tính linh hoạt về vị trí và tỷ lệ đối tượng, giúp mô hình không bị lệ thuộc vào vị trí cố định trong khung hình.

#### 4. Thêm nhiễu Gaussian (Additive Gaussian Noise)

* Mục tiêu: Giúp mô hình học được tính chất bền vững (robustness) trước các ảnh chứa nhiễu, giống như trong môi trường thực tế với ảnh từ camera chất lượng thấp.
* Phương pháp: Thêm nhiễu phân phối Gaussian với trung bình bằng 0 và phương sai nằm trong khoảng đã định (ví dụ: 10–30).
* Tác động: Mô hình không bị phụ thuộc vào chi tiết nhỏ lẻ trong ảnh, từ đó tránh bị nhiễu hóa ảnh hưởng đến kết quả dự đoán.

#### 5. Chuyển sang ảnh xám (Grayscale Conversion)

* Mục tiêu: Đơn giản hóa biểu diễn ảnh, đặc biệt hữu ích trong bài toán nhận diện hình dạng hoặc biên (edge-based tasks).
* Phương pháp: Sử dụng công thức chuyển đổi RGB sang grayscale, giảm ảnh từ 3 kênh màu còn 1 kênh sáng.
* Tác động: Giảm thiểu độ phức tạp của dữ liệu, làm tăng tốc độ xử lý và tiết kiệm tài nguyên tính toán, đồng thời vẫn giữ được thông tin cấu trúc chính.

6. Tách nền (Background Removal)

* Mục tiêu: Tập trung vào đối tượng chính, loại bỏ các yếu tố gây nhiễu từ nền ảnh.
* Phương pháp: Áp dụng thuật toán GrabCut để phân vùng ảnh thành vùng foreground (đối tượng chính) và background, sau đó loại bỏ phần nền không cần thiết.
* Tác động: Cải thiện đáng kể hiệu quả của mô hình trong các bài toán như nhận diện sản phẩm, phân loại hình ảnh, nơi đối tượng chính chiếm phần nhỏ trong ảnh.

#### 7. Trích xuất biên (Edge Detection)

* Mục tiêu: Nhấn mạnh vào hình dạng và cấu trúc của đối tượng.
* Phương pháp: Sử dụng thuật toán Canny Edge Detector để phát hiện biên – những điểm có độ biến thiên cường độ cao trong ảnh.
* Tác động: Tạo ra một dạng biểu diễn khác của dữ liệu ảnh tập trung vào đặc trưng hình học, hữu ích cho các bài toán phát hiện vật thể hoặc phân đoạn ảnh.

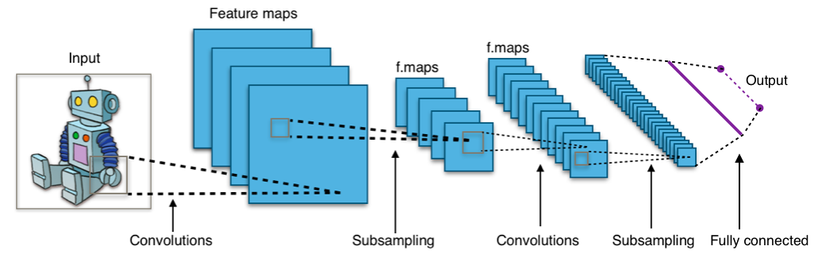
Việc kết hợp đồng thời nhiều kỹ thuật biến đổi ảnh giúp tạo ra một tập dữ liệu đa dạng từ nguồn ảnh ban đầu, đáp ứng yêu cầu thực tế và làm tăng khả năng khái quát hóa của mô hình học máy. Không chỉ dừng lại ở việc "sao chép và biến dạng" ảnh gốc, các kỹ thuật này còn đóng vai trò như một phương pháp giả lập dữ liệu (data simulation) nhằm huấn luyện mô hình trong điều kiện giả lập trước khi triển khai trong môi trường thật.

### **2.2.2. Kiến trúc mô hình CNN**

*2.2.2.1. Kiến trúc mạng CNN*

Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng(training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel >edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh



Hình 2.2 Kiến trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên. Mạng CNN sử dụng 3 ý tưởng cơ bản:

* ·Các trường tiếp nhận cục bộ (local receptive field)
* ·Trọng số chia sẻ (shared weights)
* ·Tổng hợp (pooling).

***2.2.2. Các lớp cơ bản và nguyên lý hoạt động của CNN***

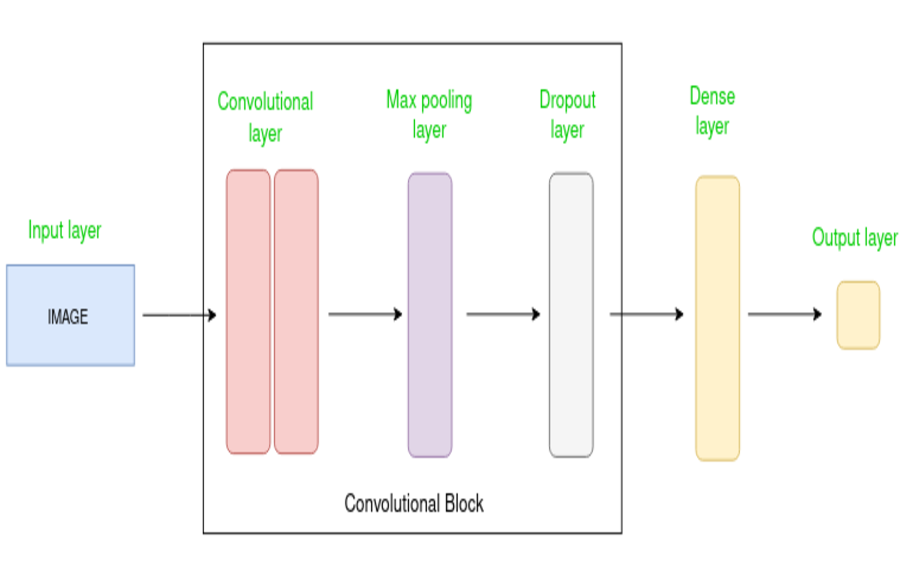
**Convolutional layer**

Convolutional layer là lớp tích chập là thành phần quan trọng nhất của CNN, chịu trách nhiệm trích xuất các đặc trưng từ dữ liệu đầu vào. Lớp này sử dụng một bộ lọc (kernel) - một ma trận nhỏ có kích thước phổ biến như 3x3 hoặc 5x5 - quét qua từng vùng nhỏ của hình ảnh và thực hiện phép nhân tích chập (convolution) giữa các giá trị pixel với trọng số của bộ lọc. Kết quả của quá trình này tạo thành bản đồ đặc trưng (feature map), giúp mô hình phát hiện các đặc điểm như cạnh, góc, màu sắc hoặc kết cấu trong ảnh.

Các tham số quan trọng của lớp tích chập bao gồm: Số lượng bộ lọc, Stride (bước di chuyển của bộ lọc) và Padding (giữ kích thước ảnh). Trong đó:

* Stride xác định khoảng cách di chuyển của kernel trên ảnh đầu vào theo cả chiều ngang (trái sang phải) và chiều dọc (trên xuống dưới).
* Padding là quá trình thêm giá trị vào viền ảnh để kiểm soát kích thước feature map, bảo vệ thông tin viền ảnh khi thực hiện tích chập.

Sau mỗi phép tích chập, Convolutional Neural Networks thường áp dụng hàm kích hoạt ReLU (Rectified Linear Unit) để loại bỏ giá trị âm, tăng tính phi tuyến và giúp mô hình học hiệu quả hơn.



Hình 2.3 Convolutional layer

Convolutional layer chịu trách nhiệm trích xuất các đặc trưng từ dữ liệu đầu vào

**Pooling layer**

Sau khi trích xuất đặc trưng qua lớp tích chập, Convolutional Neural Networks sử dụng Pooling Layer để giảm kích thước feature map, từ đó giảm số lượng tham số, tăng hiệu suất tính toán và tránh hiện tượng overfitting (mô hình học quá kỹ vào dữ liệu huấn luyện, nhưng lại hoạt động kém khi gặp dữ liệu mới). Pooling hoạt động bằng cách áp dụng một bộ lọc nhỏ (thường là 2x2 hoặc 3x3) để lấy giá trị đại diện cho mỗi vùng quét, giúp giữ lại những thông tin quan trọng nhất.

Có hai phương pháp pooling phổ biến: Max Pooling và Average Pooling.

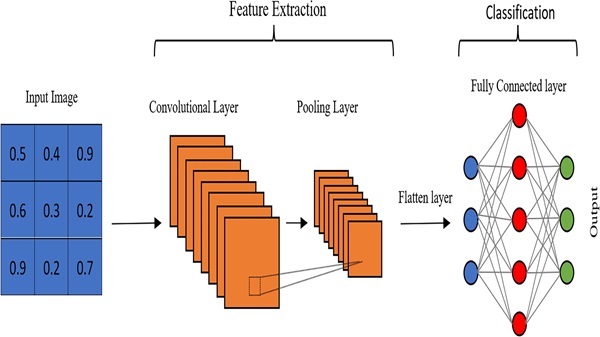
* Trong Max Pooling, giá trị lớn nhất trong vùng quét sẽ được giữ lại, giúp mô hình tập trung vào những đặc trưng nổi bật nhất.
* Average Pooling tính trung bình các giá trị trong vùng quét, giúp tổng hợp thông tin thay vì chỉ giữ giá trị lớn nhất như Max Pooling.

Mặc dù pooling làm mất đi một số thông tin, nhưng đổi lại, nó giúp mô hình hoạt động hiệu quả hơn, giảm thiểu độ phức tạp và cải thiện khả năng tổng quát hóa đối với dữ liệu mới.

**Fully connected layer**

Fully connected layer là lớp kết nối đầy đủ nằm ở cuối mạng Convolutional Neural Networks, đóng vai trò tổng hợp tất cả các đặc trưng đã trích xuất và thực hiện nhiệm vụ phân loại hình ảnh. Ở lớp này, mỗi nơ-ron được kết nối với toàn bộ nơ-ron ở lớp trước, tạo nên một mạng lưới liên kết chặt chẽ. Các giá trị từ feature map trước đó sẽ được chuyển thành một vector một chiều, một chuỗi dài duy nhất và đưa vào lớp fully connected để xử lý. Quá trình này được gọi là Làm phẳng Flattening.

Tiếp đó, CNN sử dụng các hàm kích hoạt phi tuyến như Softmax hoặc Sigmoid để tính toán xác suất cho từng lớp đầu ra. Điều này giúp cho mô hình đưa ra quyết định cuối cùng, chẳng hạn như phân loại hình ảnh thành các nhóm khác nhau (ví dụ: chó, mèo, ô tô, v.v.).



Hình 2.4 Fully connected layer

Fully connected layer tổng hợp tất cả các đặc trưng đã trích xuất và thực hiện nhiệm vụ phân loại hình ảnh

**Additional layers**

Bên cạnh ba lớp chính, CNN có thể bao gồm một số lớp bổ sung để tối ưu hiệu suất và độ chính xác của mô hình.

* Lớp kích hoạt (Activation Layer) giúp tăng khả năng học và tạo tính phi tuyến cho mô hình bằng cách áp dụng các hàm phi tuyến như ReLU, Sigmoid hoặc Tanh.
* Lớp dropout (Dropout Layer) là một kỹ thuật quan trọng để giảm overfitting, bằng cách tạm thời loại bỏ ngẫu nhiên một số nơ-ron trong quá trình huấn luyện, giúp mô hình trở nên linh hoạt và tổng quát hơn.
* Ngoài ra, lớp chuẩn hóa (Batch Normalization Layer) cũng thường được sử dụng để tăng tốc độ huấn luyện và giúp mô hình ổn định hơn bằng cách chuẩn hóa dữ liệu giữa các lớp.

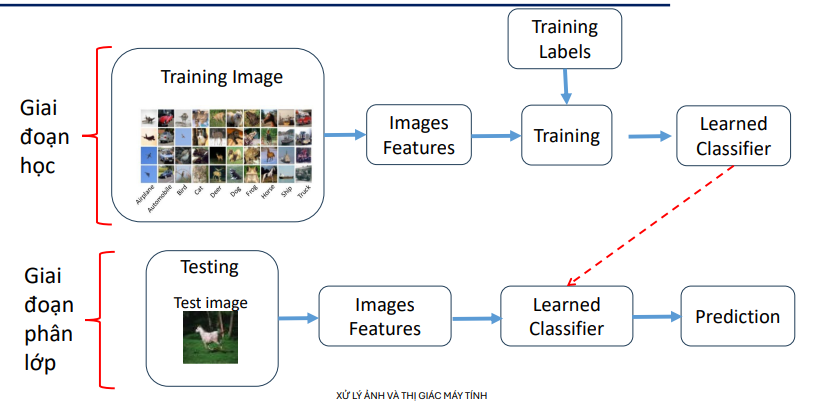
### **2.2.3. Cách chọn tham số cho CNN**

Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể

Filter size: thường filter theo size 5×5 hoặc 3×3

Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn

Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất



Hình 2.5 Mô hình quy trình bài toán

### **2.2.4. Chi tiết huấn luyện**

* Chia tập dữ liệu thành:
* Train: 70%
* Validation: 15%
* Test: 15%

1. ***Dữ liệu và tiền xử lý***

Mô hình được huấn luyện trên tập dữ liệu đã được chia thành ba phần: train, val, và test. Kích thước ảnh được chuẩn hóa về 256x256, và các ảnh được chuẩn hóa giá trị điểm ảnh về khoảng [0, 1].

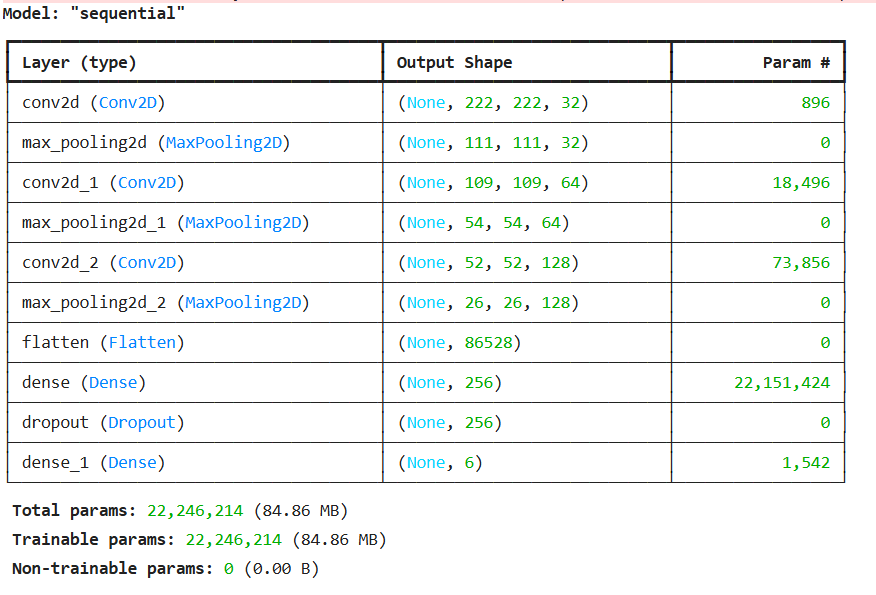
Để cải thiện khả năng tổng quát hóa, dữ liệu huấn luyện được áp dụng một số kỹ thuật tăng cường dữ liệu (data augmentation) như:

* Xoay ảnh ngẫu nhiên
* Dịch chuyển chiều rộng, chiều cao
* Zoom và lật ngang ảnh
* Thêm 1 số phương pháp nữa của Lộc nêu ở trên (Trích chọn đặc trưng, xử lý nhiễu Gauss,....)

1. ***Kiến trúc mô hình***

Mô hình sử dụng kiến trúc mạng nơ-ron tích chập (CNN) đơn giản gồm:

* 3 tầng Conv2D kết hợp MaxPooling2D
* 1 tầng Flatten, tiếp theo là Dense với 256 nút và Dropout để giảm overfitting
* Tầng cuối là Dense(6) với hàm kích hoạt softmax tương ứng với 6 lớp bệnh



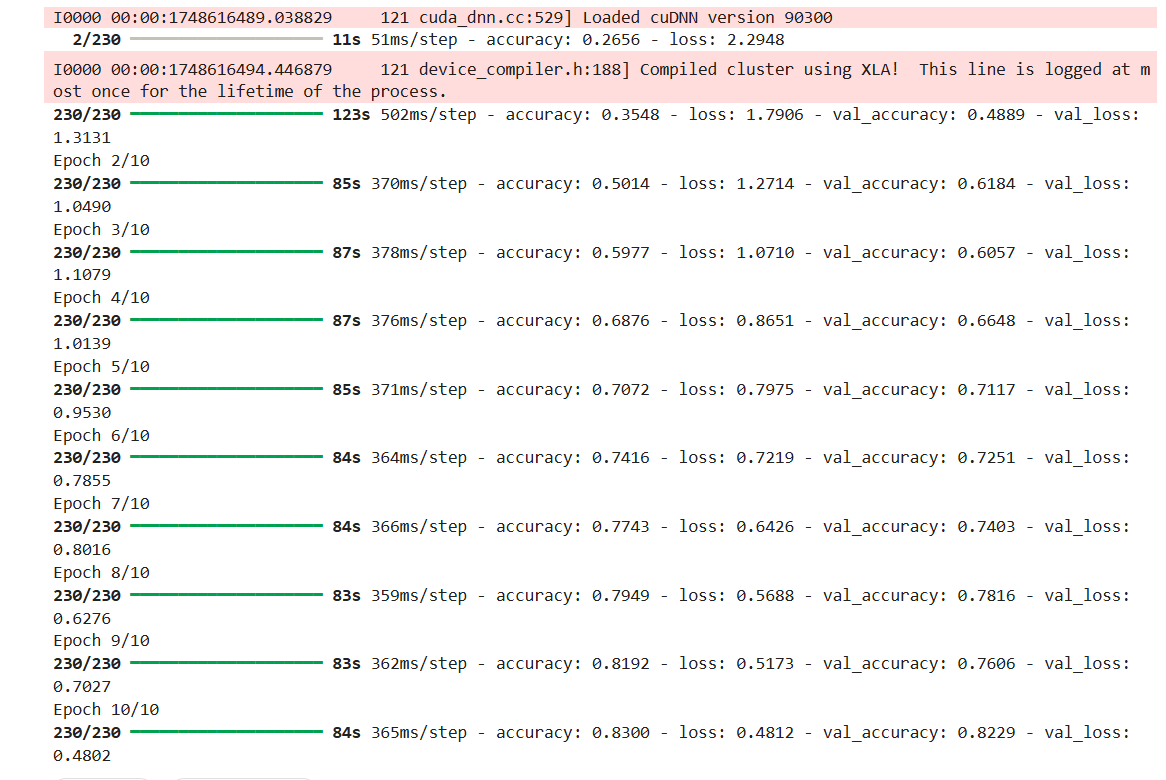
Hình 2.6 Kiến trúc mạng nơ-ron tích chập



Hình 2.7 Sơ đồ kiến trúc của mô hình CNN sử dụng trong huấn luyện

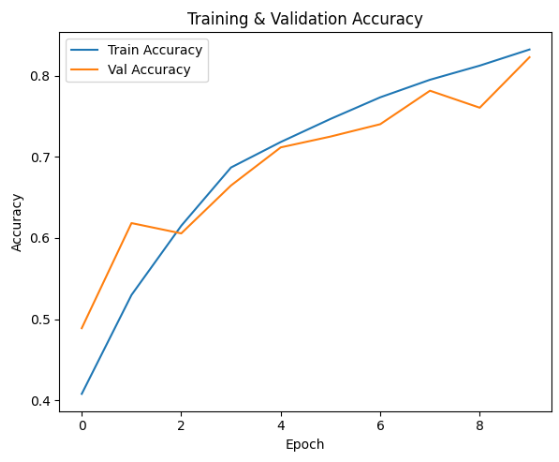
1. Quá trình huấn luyện

Mô hình được huấn luyện trong **10 epoch** với optimizer=Adam và loss=categorical\_crossentropy.

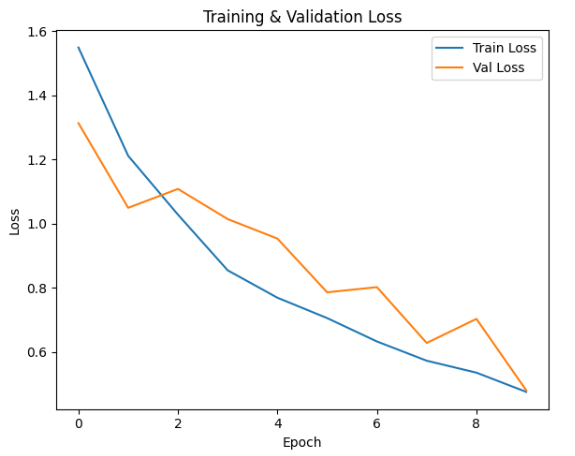


Hình 2.8 Quá trình huấn luyện

Biểu đồ dưới đây thể hiện sự thay đổi của độ chính xác và hàm mất mát trên tập huấn luyện và validation qua từng epoch.



Hình 2.9 Biểu đồ Accuracy theo từng epoch



Hình 2.10 Biểu đồ Loss theo từng epoch

1. Kết quả

* **Độ chính xác trên tập validation**: **~83%**
* **Độ chính xác trên tập test**: gần tương đương, cho thấy mô hình không bị overfit nghiêm trọng

→ Đây là một kết quả ổn với mô hình CNN đơn giản, phù hợp với bài toán

1. Lưu mô hình

* Kết quả cuối cùng sau quá trình huấn luyện sẽ được 1 file lưu dưới định dạng .keras để thuận tiện cho việc tái sử dụng, hoặc tiếp tục huấn luyện sau này.

├── dataset/

│ ├── train/

│ │ ├── dom\_la/ (350\*3 ảnh)

│ │ ├── dom\_vi\_khuan/ (350\*3)

│ │ ├── moc\_suong/ (350\*3)

│ │ ├── nam/ (350\*3)

│ │ ├── phan\_trang/ (350\*3)

│ │ └── healthy/ (700\*3)

│ ├── val/

│ │ ├── dom\_la/ (75\*3 ảnh)

│ │ ├── dom\_vi\_khuan/ (75\*3)

│ │ ├── moc\_suong/ (75\*3)

│ │ ├── nam/ (75\*3)

│ │ ├── phan\_trang/ (75\*3)

│ │ └── healthy/ (150\*3)

│ └── test/

│ ├── dom\_la/ (75\*3 ảnh)

│ ├── dom\_vi\_khuan/ (75\*3)

│ ├── moc\_suong/ (75\*3)

│ ├── nam/ (75\*3)

│ ├── phan\_trang/ (75\*3)

│ └── healthy/ (150\*3)

Hình 2.11 Cấu trúc thư mục

*\*Chú ý:* Trong các tập train, test, val đã bao gồm cả các dữ liệu sau khi làm giàu (Grayscale, Segmented)

* Và class\_indices.json, nhằm ánh xạ chỉ số đầu ra của mô hình với tên lớp tương ứng (Các bệnh)

{

"0":"dom\_la",

"1":"dom\_vi\_khuan",

"2":"healthy",

"3":"moc\_suong",

"4":"nam",

"5":"phan\_trang"

}

## **2.4. Công cụ, môi trường phát triển**

### **2.4.1. Môi trường huấn luyện**

Quá trình huấn luyện mô hình được thực hiện trên nền tảng **Kaggle**, một môi trường điện toán đám mây miễn phí, hỗ trợ **GPU hiệu suất cao** (GPU T4x2, GPU P100) giúp tăng tốc quá trình huấn luyện mô hình học sâu.

Trong khi đó, nếu huấn luyện trên máy tính cá nhân, thường không có GPU chuyên dụng hoặc chỉ có GPU yếu, dẫn đến thời gian huấn luyện kéo dài và không khả thi với mô hình lớn.

Một số công cụ, thư viện được sử dụng trong huấn luyện:

Bảng 2.1 Công cụ thư viện sử dụng

|  |  |
| --- | --- |
| Ngôn ngữ lập trình | Python3 |
| TensorFlow | Xây dựng và huấn luyện mô hình CNN |
| Keras (Tích hợp trong TensorFlow) | Tổ chức mô hình theo dạng Sequential |
| Numpy | Mô hình toán học mạnh mẽ, xử lý mảng số |
| Matplotlib | Vẽ biểu đồ trực quan hóa quá trình học |
| ImageDataGenerator | Tăng cường dữ liệu → Mô hình tổng quát tốt hơn |
| json | Lưu ánh xạ nhãn lớp vào file .json (class\_indices.json) |

### **2.4.2. Giao diện người dùng**

Ứng dụng được triển khai trên giao diện một website để người dùng có thể tải ảnh lên và nhận kết quả chẩn đoán bệnh.

Ngôn ngữ sử dụng:

Bảng 2.2 Ngôn ngữ sử dụng cho giao diện

|  |  |
| --- | --- |
| HTML | Tạo cấu trúc giao diện |
| CSS | Thiết kế giao diện đơn giản, dễ nhìn |
| JAVASCRIPT | Gửi dữ liệu ảnh và nhận kết quả dự đoán từ backend bằng API |

### **2.4.3. Xử lý phía máy chủ**

Được xây dựng bằng **Flask**, một micro web framework gọn nhẹ của Python, cho phép triển khai nhanh REST API.

Chức năng:

* Nhận ảnh từ người dùng
* Dự đoán nhãn bệnh trên lá với mô hình đã huấn luyện (.keras)
* Trả kết quả dự đoán về frontend dưới dạng JSON

Một số thư viện được sử dụng trong Backend gồm:

Bảng 2.3 Một số thư viện sử dụng trong Backend

|  |  |
| --- | --- |
| Flask | Tạo server và định tuyến API |
| TensorFlow | Tải và sử dụng mô hình .keras đã huấn luyện |
| PIL (Pillow) | Xử lý và resize ảnh đầu vào |
| NumPy | Chuyển đổi ảnh về mảng số để đưa vào mô hình |
| Os | Quản lý đường dẫn và lưu trữ ảnh |
| json | Đọc file ánh xạ lớp (class\_indices.json) |

## **2.5. Triển khai mô hình lên Website**

Để tăng tính ứng dụng thực tế của hệ thống, nhóm đã tiến hành triển khai mô hình nhận diện bệnh lá cây lên một website đơn giản. Website cho phép người dùng tải ảnh lá cây cần chẩn đoán, sau đó hệ thống sẽ xử lý và hiển thị kết quả nhận diện bệnh tương ứng.

### **2.5.1. Kiến trúc hệ thống**

Hệ thống được tổ chức theo các khối gồm:

User: Người dùng sử dụng trình duyệt để tải ảnh lên hệ thống.

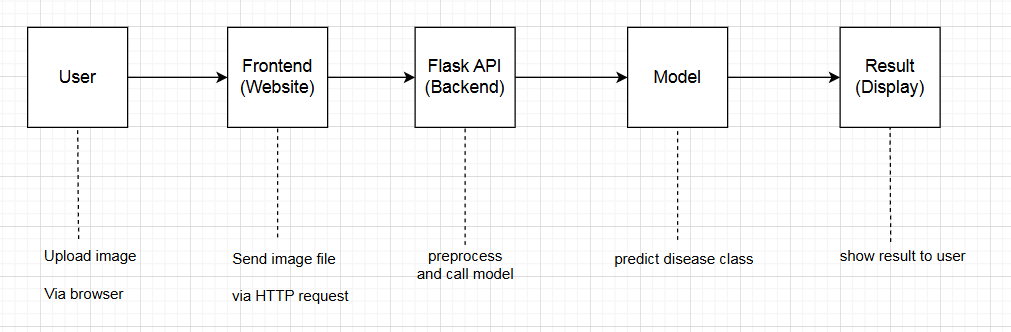
Frontend: Hiển thị giao diện, cho phép người dùng truy cập website tải ảnh từ máy tính để phân tích và hiển thị kết quả dự đoán.

Backend (Flash API): tiếp nhận ảnh, gọi model để xử lý và hiển thị kết quả

Model: mô hình học sâu được được huấn luyện và lưu ở định dạng .h5 và file ánh xạ bệnh định dạng .json

Result: Kết quả dự đoán hiển thị rõ ràng cho người dùng ngay trên website.

Kiến trúc tổng thể hệ thống:



Hình 2.12 Quy trình xử lý phía máy chủ

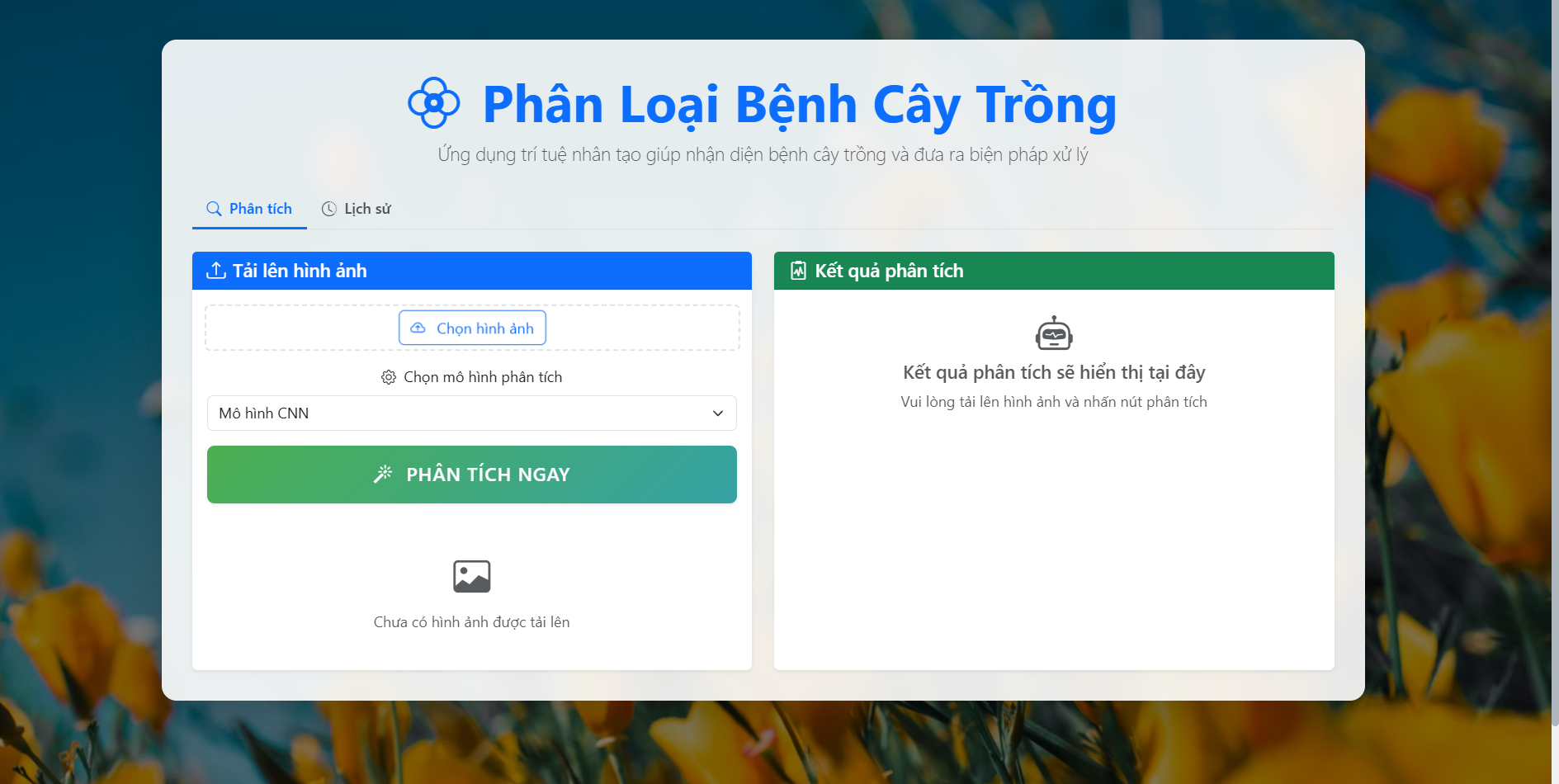
### **2.5.2. Quy trình hoạt động**

Khi người dùng tải ảnh lên giao diện web, ảnh sẽ được gửi đến máy chủ thông qua API /predict do Flask cung cấp. Backend Flask sẽ tiếp nhận ảnh, thực hiện các bước tiền xử lý (resize về kích thước 224x224, chuẩn hóa ảnh) rồi đưa ảnh vào mô hình học sâu đã huấn luyện (firstest.h5). Dựa trên kết quả dự đoán (vector xác suất), hệ thống xác định chỉ số lớp có xác suất cao nhất, sau đó ánh xạ sang nhãn bệnh tương ứng thông qua file class\_indices.json. Kết quả nhận diện sau cùng được phản hồi về frontend và hiển thị trực tiếp cho người dùng.

### **2.5.3. Giao diện người dùng**

Giao diện web được thiết kế đơn giản, thân thiện, tập trung vào trải nghiệm người dùng:

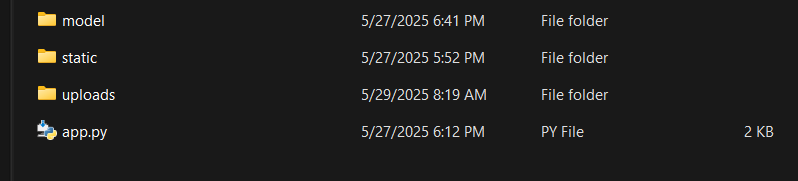
* Giao diện chính gồm: nút tải ảnh, vùng xem trước ảnh và khu vực hiển thị kết quả.
* Kết quả trả về rõ ràng, dễ hiểu, giúp người dùng nhanh chóng nắm bắt được thông tin về loại bệnh cây trồng.



Hình 2.13 Giao diện của mô hình khi được triển khai

### **2.5.4. Tổ chức dự án**

Toàn bộ hệ thống được tổ chức gọn gàng theo thư mục sau:



Hình 2.14 Cấu trúc dự án

* model/ chứa mô hình và file ánh xạ nhãn.
* uploads/ lưu tạm thời ảnh do người dùng tải lên.
* static/ chứa các tệp HTML, CSS, JavaScript phục vụ frontend.
* app.py là tệp chính điều phối Flask API và gọi mô hình dự đoán.

Cách tổ chức này đảm bảo việc duy trì, phát triển, mở rộng hoặc triển khai thực tế được thuận tiện về sau.

# CHƯƠNG III: THỰC NGHIỆM

* 1. **Dữ liệu** 
     1. ***Nguồn dữ liệu***

Dữ liệu ảnh lá cây dùng để huấn luyện và kiểm tra mô hình chủ yếu được nhóm nghiên cứu, thu thập từ nguồn công khai trên Kaggle có chọn lọc, chiếm khoảng 90% tổng dữ liệu bao gồm 5 loại bệnh phổ biến (Ảnh màu). Ngoài ra, nhóm còn tự thu thập thêm khoảng 10% dữ liệu bằng cách chụp ảnh lá cây thực tế tại vườn nhà, giúp tăng tính đa dạng và độ thực tế cho bộ dữ liệu.

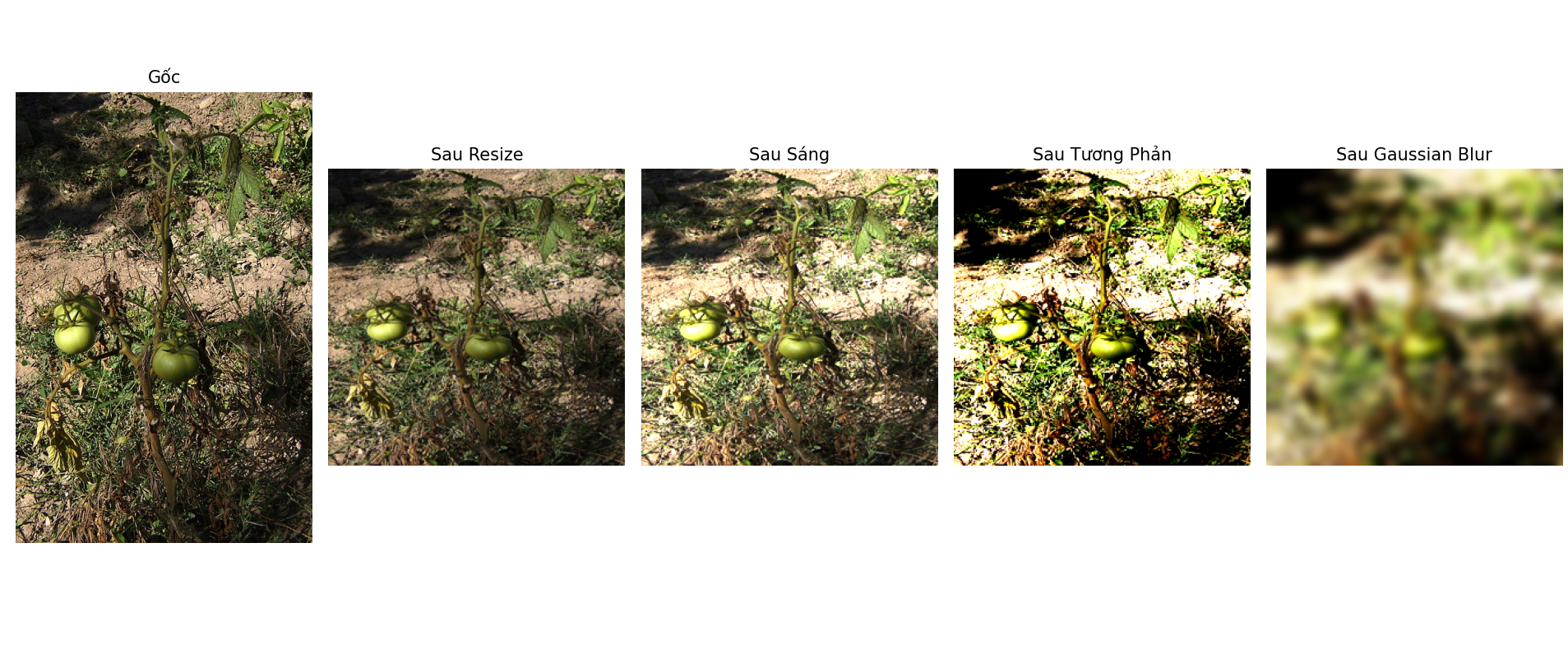
### **3.1.2. Tiền xử lý**

Để đảm bảo chất lượng đầu vào và tăng hiệu quả huấn luyện mô hình, dữ liệu ảnh ban đầu được xử lý qua một loạt các bước tiền xử lý, giúp chuẩn hóa, làm giàu thông tin và đa dạng hóa bộ dữ liệu.

Quy trình tiền xử lý trải qua 3 giai đoạn:

Chuẩn hóa ảnh

* **Resize**: Tất cả ảnh được resize về cùng một kích thước cố định (256x256 pixel) nhằm phù hợp với yêu cầu đầu vào của mô hình CNN.
* **Lọc nhiễu** (Noise Filtering): Giảm ảnh hưởng của nhiễu và chi tiết không quan trọng. Sử dụng Gaussian Blur để làm mượt vùng ảnh đồng thời giữ lại biên cạnh quan trọng
* **Cân bằng độ sáng/ tương phản:** Cải thiện chất lượng thị giác và tăng cường chi tiết vùng tối/sáng



Hình 3.1 ảnh sau khi xử lý

→ **Không làm tăng số lượng ảnh**, mà chỉ **cải thiện chất lượng của ảnh gốc**.

* + 1. ***Làm giàu dữ liệu***

Data Augmentation là quá trình làm tăng số lượng ảnh huấn luyện bằng cách sinh ra các biến thể khác nhau từ ảnh gốc. Các kỹ thuật áp dụng bao gồm:

Bảng 3.1 Các kĩ thuật làm giàu dữ liệu

|  |  |
| --- | --- |
| **Kỹ thuật** | **Mô tả** |
| Xoay ảnh (Rotate) | Xoay ảnh trong khoảng [-30°, +30°] để mô phỏng các góc nhìn khác nhau. |
| Thay đổi độ tương phản/sáng | Điều chỉnh ánh sáng/tương phản ảnh để mô phỏng điều kiện ánh sáng khác nhau. |
| Zoom hoặc crop | Phóng to hoặc cắt ảnh ngẫu nhiên để tăng tính đa dạng vùng tập trung. |
| Thêm nhiễu Gaussian | Thêm nhiễu Gaussian giúp mô hình kháng nhiễu tốt hơn. |
| Chuyển sang ảnh xám | Biến ảnh RGB thành ảnh đen trắng để huấn luyện các mô hình đơn kênh. |
| Tách nền | Sử dụng GrabCut để tách đối tượng chính khỏi nền ảnh. |
| Trích xuất biên | Dùng Canny Edge Detection để lấy thông tin biên trong ảnh. |

A collage of photos of a leaf

AI-generated content may be incorrect.  
Hình 3.2 Kết quả ảnh được làm giàu

Cấu trúc thư mục sau khi làm giàu dữ liệu

images/ # Thư mục chứa ảnh gốc

augmented/

├── ten\_anh1/

│ ├── rgb.jpg

│ ├── rotated.jpg

│ ├── bright\_contrast.jpg

│ ├── zoomed.jpg

│ ├── noisy.jpg

│ ├── gray.jpg

│ ├── edges.jpg

│ └── segmented.jpg

├── ten\_anh2/

│ └── ...

**Kết quả đạt được:**

* Hệ thống đã xử lý toàn bộ ảnh trong thư mục đầu vào và sinh ra các biến thể khác nhau.
* Số lượng ảnh tăng lên gấp **7 lần**, tùy theo số kỹ thuật áp dụng.
* Tập dữ liệu phong phú hơn, thuận tiện cho bước huấn luyện mô hình phân loại hoặc nhận diện.

### **3.1.4. Phân đoạn ảnh và đặc trưng**

**Phát hiện biên bằng phương pháp Canny**

Đầu tiên, ảnh lá cây được chuyển sang dạng ảnh xám để giảm chiều dữ liệu và làm nổi bật thông tin kết cấu. Sau đó, thuật toán phát hiện biên Canny được áp dụng nhằm làm nổi rõ ranh giới các vùng trên lá – đây là những vùng có thể chứa thông tin quan trọng như rìa vết bệnh. Kết quả sau khi phát hiện biên được lưu lại dưới dạng ảnh nhị phân, đồng thời histogram của ảnh biên (với 256 mức xám) cũng được tính toán. Đây là đặc trưng phản ánh tần suất xuất hiện của các mức cường độ biên trong ảnh.

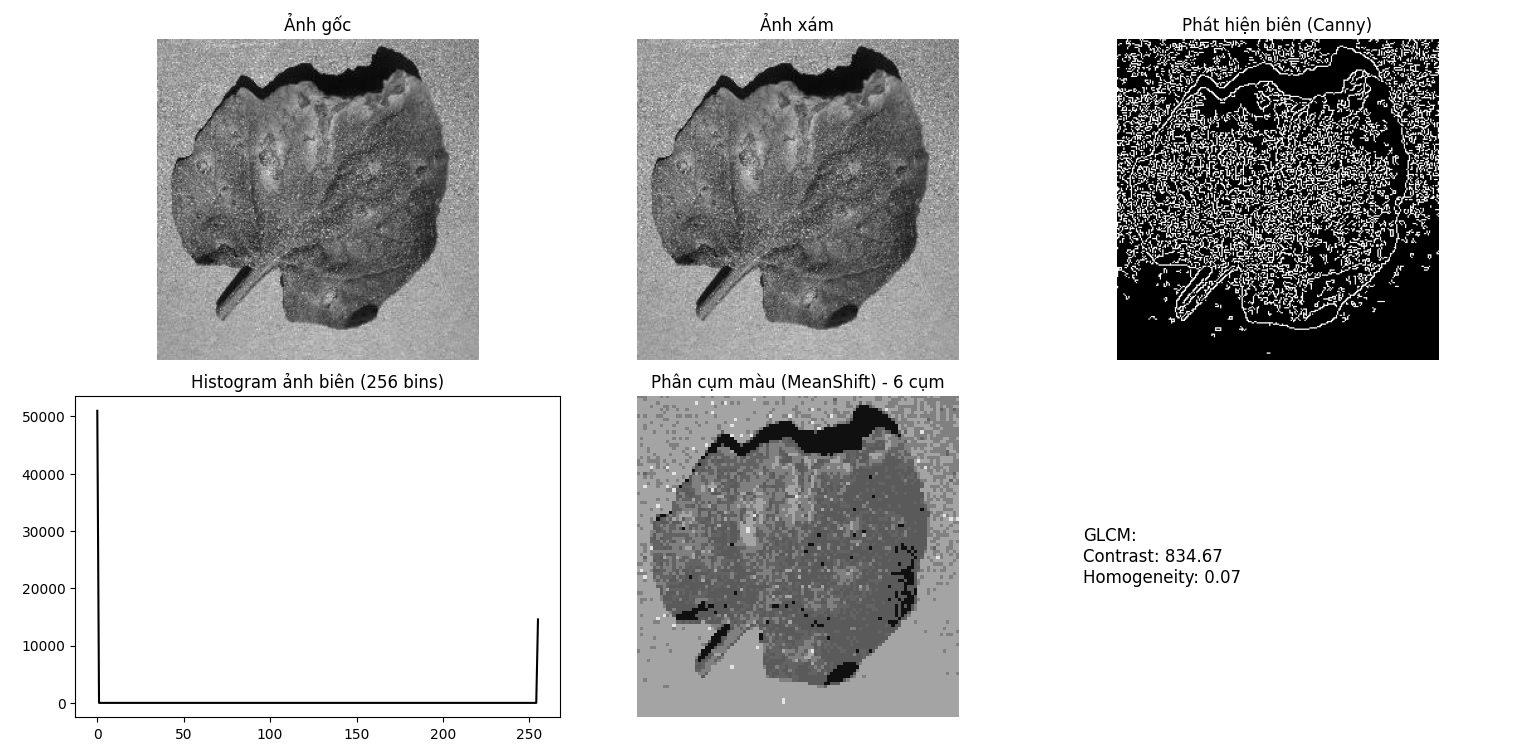
**Trích xuất đặc trưng kết cấu bằng GLCM**

Sau khi có ảnh xám, ảnh tiếp tục được phân tích cấu trúc thông qua ma trận đồng xuất hiện mức xám (Gray Level Co-occurrence Matrix – GLCM). Phương pháp này ghi nhận tần suất xuất hiện đồng thời của các cặp mức xám lân cận trong không gian ảnh. Từ GLCM, hai đặc trưng quan trọng được rút trích là:

* Contrast: phản ánh sự biến đổi cường độ trong vùng lân cận, càng cao thì ảnh càng có độ tương phản rõ nét.
* Homogeneity: đo lường mức độ đồng nhất của ảnh, giá trị cao thể hiện cấu trúc ảnh đồng đều.

**Phân cụm màu bằng thuật toán MeanShift**

Song song với xử lý kết cấu, ảnh màu ban đầu cũng được sử dụng để phân tích độ đa dạng màu sắc thông qua thuật toán phân cụm MeanShift. Để tăng hiệu quả tính toán, ảnh được giảm kích thước xuống còn 100x100 pixel. MeanShift được thực hiện sau khi ước lượng băng thông tự động, giúp chia ảnh thành các vùng màu đồng nhất. Số lượng cụm màu sau phân đoạn chính là một đặc trưng quan trọng, thể hiện sự đa dạng màu trong ảnh – một yếu tố có thể phản ánh mức độ tổn thương của lá cây.



Hình 3.3 Ảnh sau khi xử lý và phân cụm MeanShift

## **3.2. Mô hình huấn luyện**

* CNN
* MobileNetV2
* Resnet50

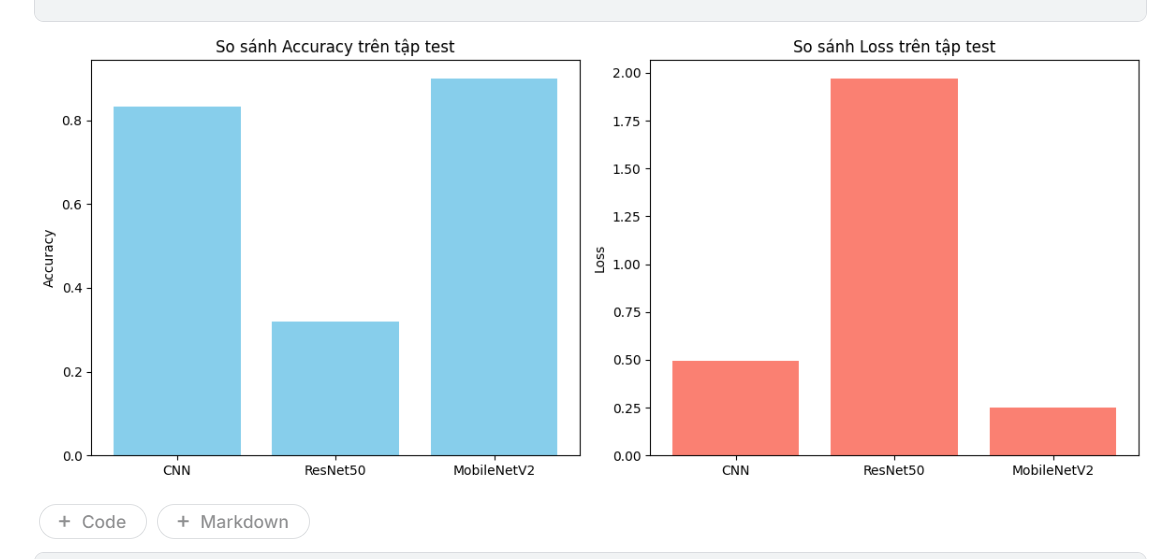
## **3.3. Kết quả thực nghiệm**

Biểu đồ Accuracy / Loss theo epoch;

|  |
| --- |
| Resnet50 |
| MobileNetV2 |
| CNN |

Hình 3.4 Biểu đồ Accuracy / Loss của Resnet50, MobileNetV2, CNN

Đánh giá mô hình:



Hình 3.5 Biểu đồ so sánh Accuracy / Loss giữa Resnet50, MobileNetV2, CNN

Bảng 3.2 Ưu, nhược điểm của các mô hình

|  |  |  |  |
| --- | --- | --- | --- |
| Đánh giá | CNN | Resnet50 | MobilenetV2 |
| Ưu điểm | Hiệu quả cao trong xử lý ảnh và nhận dạng đối tượng.  Khả năng tự động trích xuất đặc trưng (features).  Giảm số lượng tham số nhờ dùng tích chập (convolution). | Sử dụng kết nối tắt (skip connections) giúp huấn luyện mô hình sâu hơn dễ dàng.  Tránh hiện tượng biến mất gradient.  Đạt độ chính xác cao trên nhiều tác vụ thị giác máy tính. | Nhẹ, nhanh, tối ưu cho thiết bị di động và nhúng.  Sử dụng Depthwise Separable Convolution giúp giảm số tham số.  Hiệu suất tốt trên nhiều tác vụ với tài nguyên giới hạn. |
| Nhược điểm | Cần nhiều dữ liệu huấn luyện.  Tốn tài nguyên tính toán.  Dễ bị overfitting nếu mô hình lớn và dữ liệu ít. | Mô hình lớn (~25 triệu tham số), yêu cầu nhiều bộ nhớ và tài nguyên.  Thời gian suy luận chậm hơn so với các mô hình nhẹ như MobileNet. | Độ chính xác thấp hơn so với các mô hình lớn như ResNet.  Kém hiệu quả hơn trên các tác vụ phức tạp hoặc yêu cầu độ chính xác cao. |

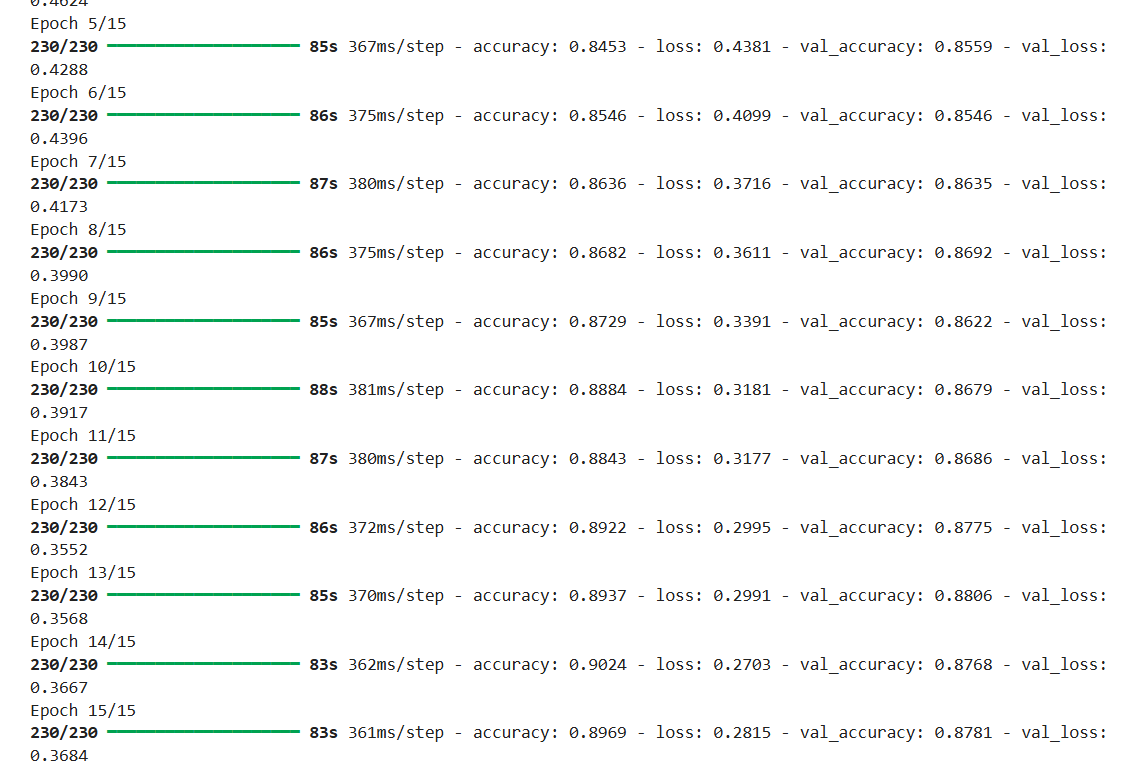
## **3.4. Đánh giá mô hình**

Mô hình sử dụng: CNN, Resnet50, MobileNetV2

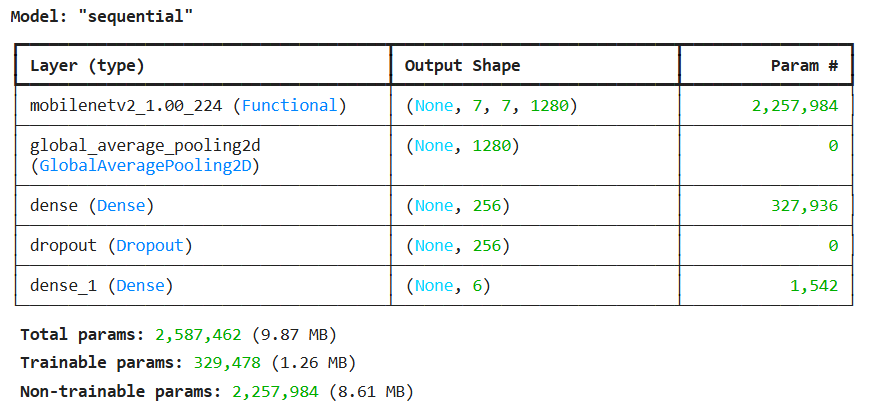
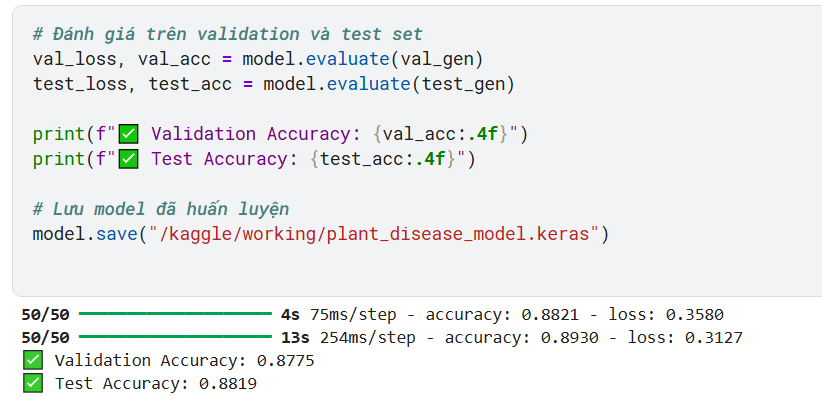
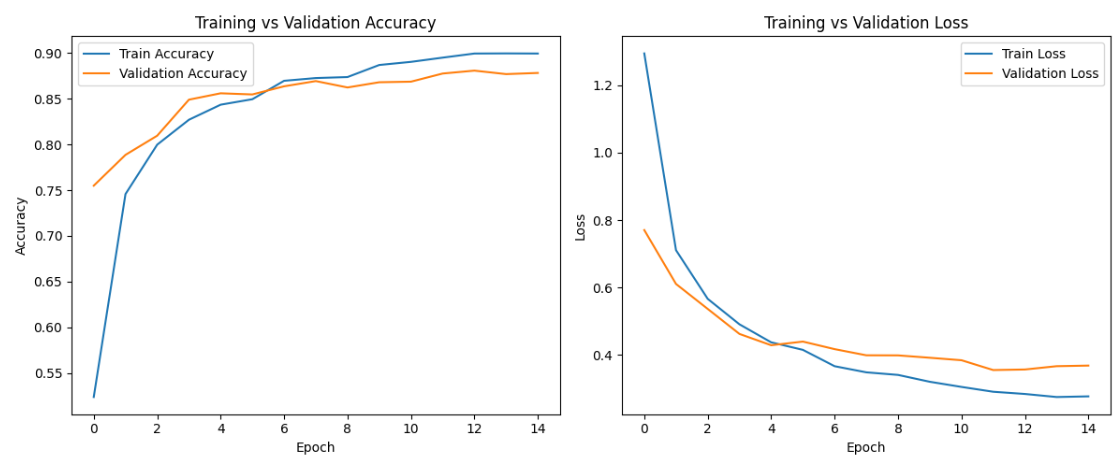
***\*CNN*** (Đã trình bày ở 2.2.4)

\*Faster R-CNN: Không dùng đc vì nó là mô hình phát hiện đối tượng (object detection) — đầu vào là ảnh, đầu ra là chính xác vùng có bệnh (bounding boxes) và nhãn của từng vùng. Mà với bài toán của nhóm thì không gán bounding boxes

***\*MobileNetV2***



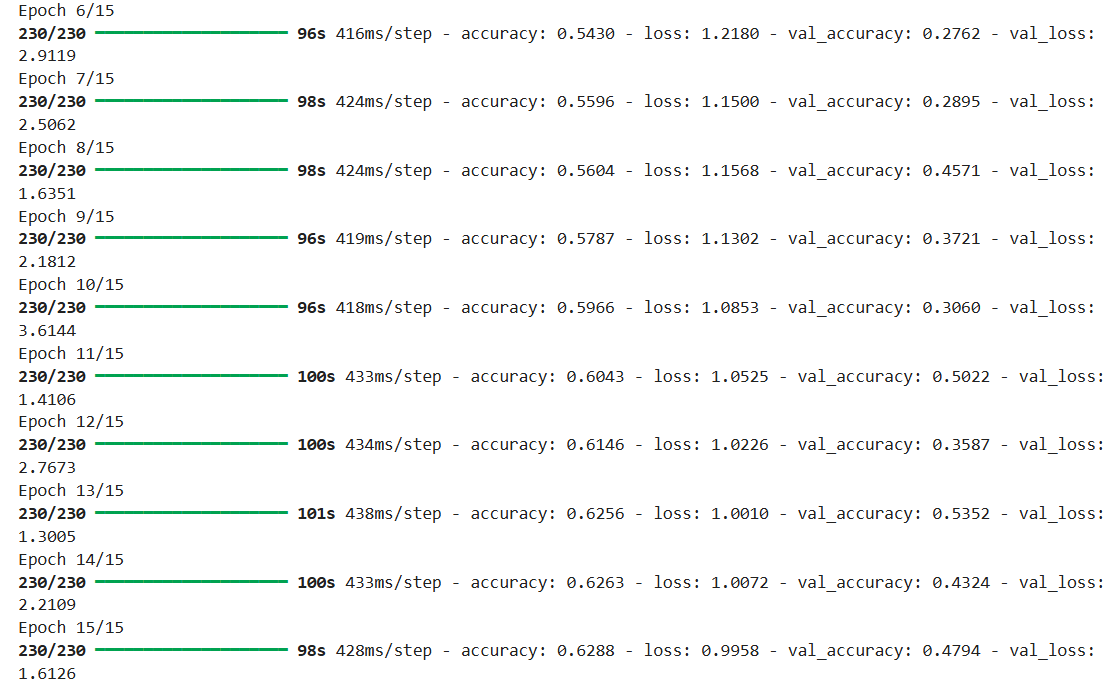
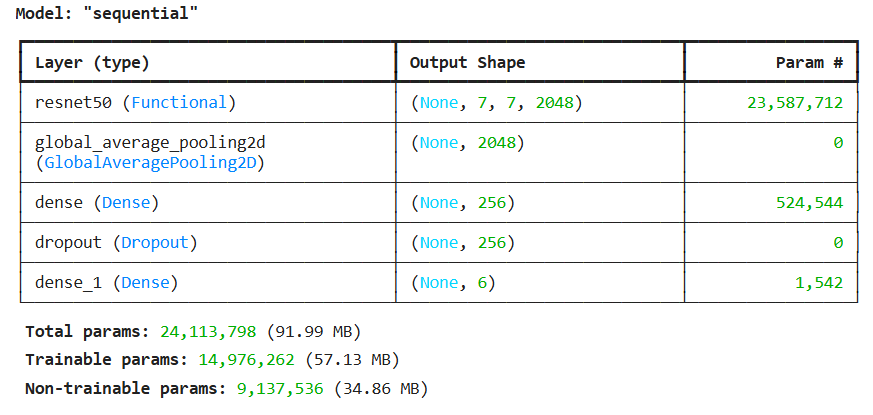
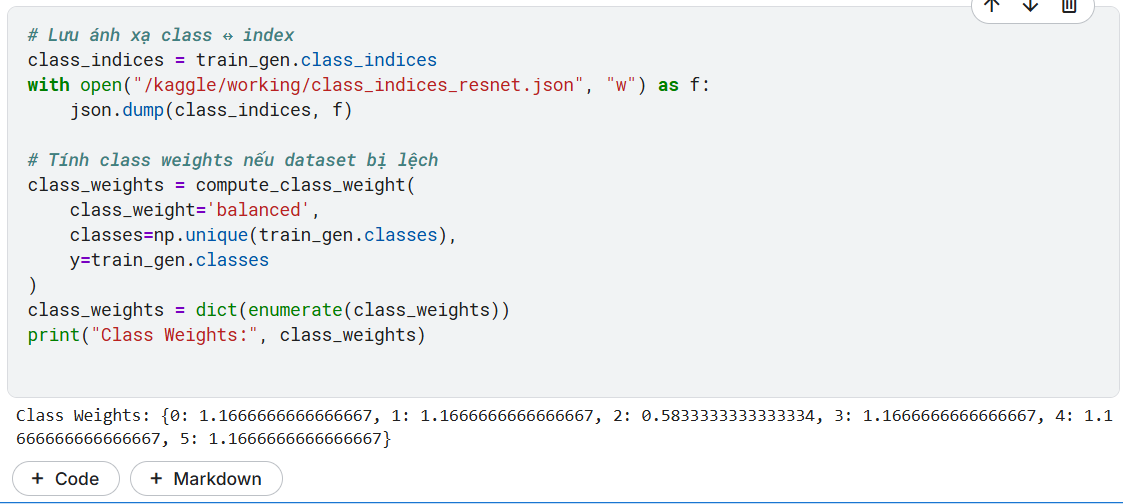
Hình 3.6 Quy trình huấn luyện mô hình MobileNetV2



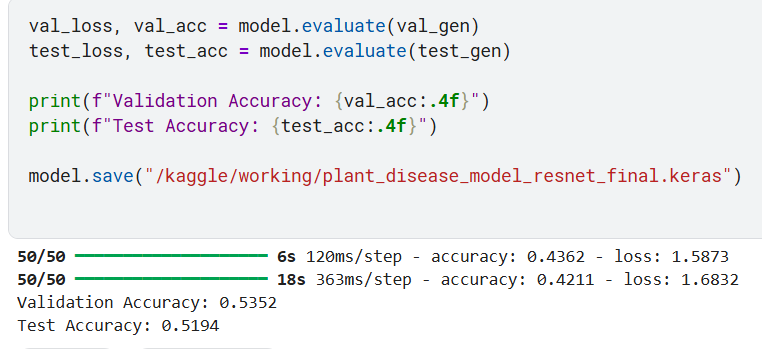
Hình 3.7 Kết quả huấn luyện mô hình MobileNetV2

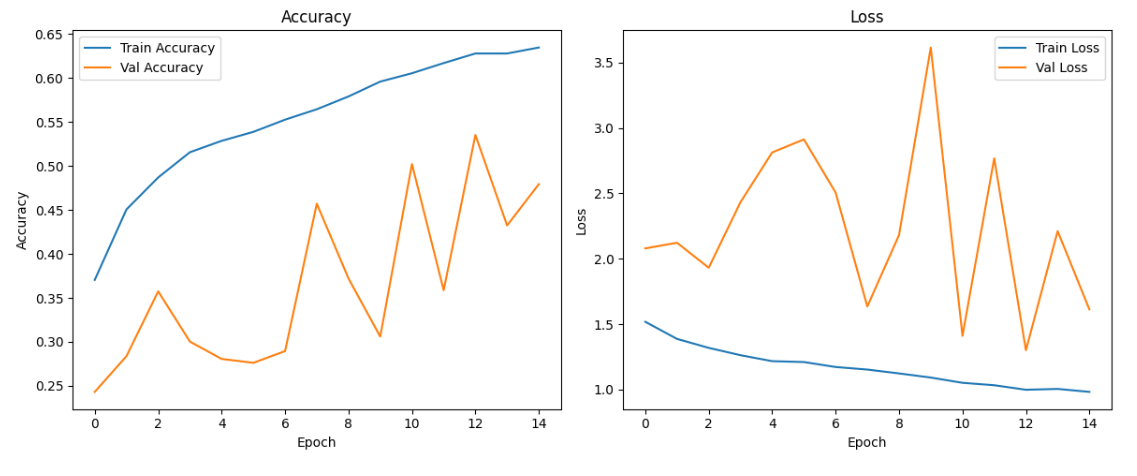
***\*Resnet50***

Quá trình huấn luyện: Mô hình được chia ra làm 15 epoch với…

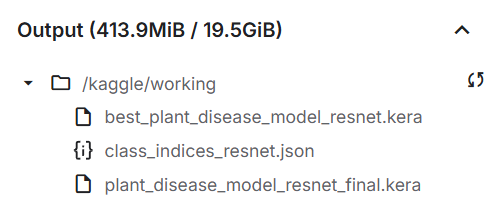


Hình 3.8 Quy trình huấn luyện mô hình Resnet50

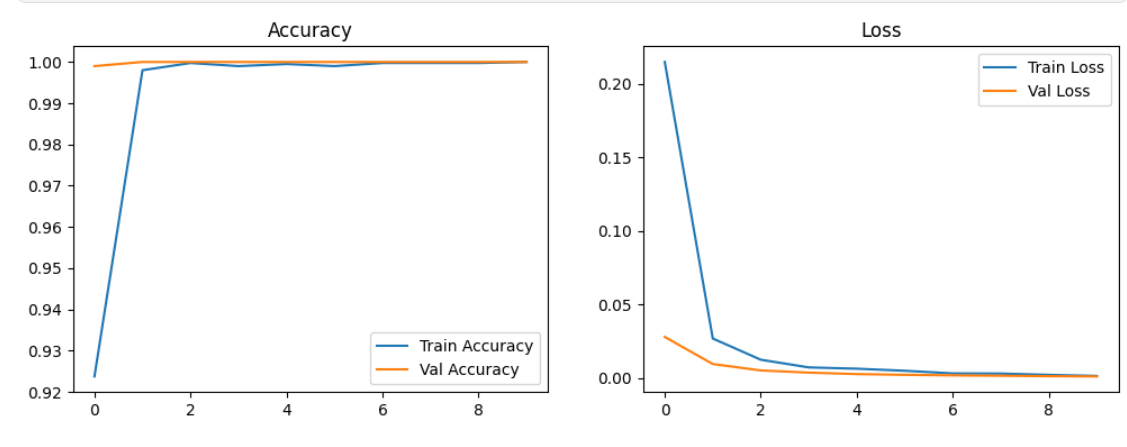
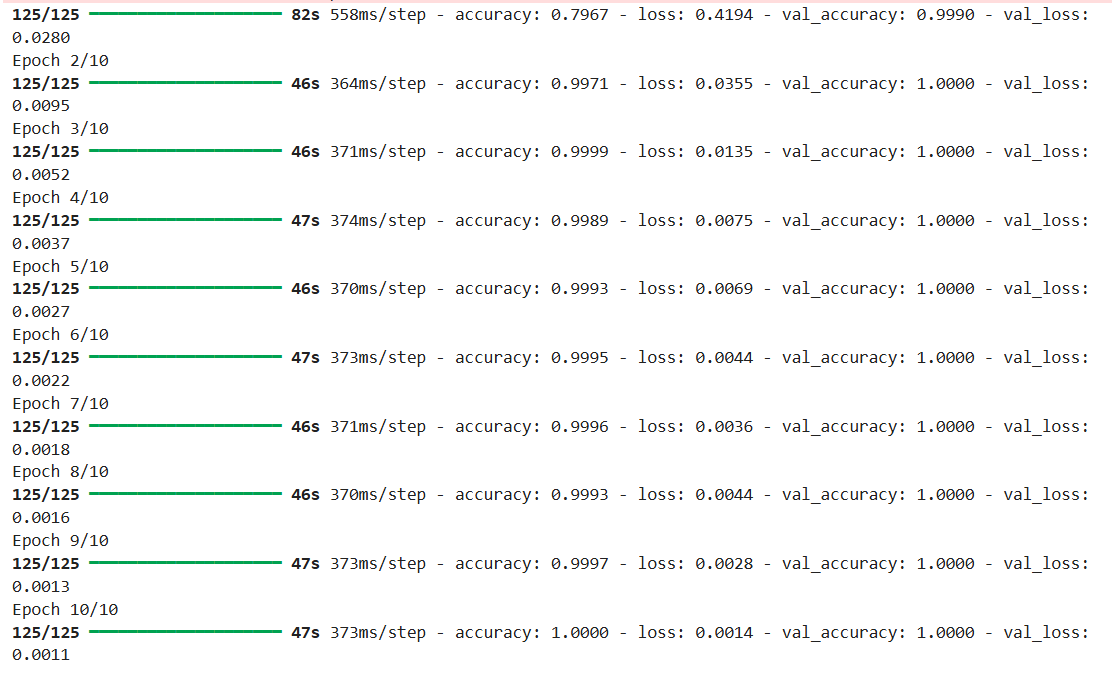
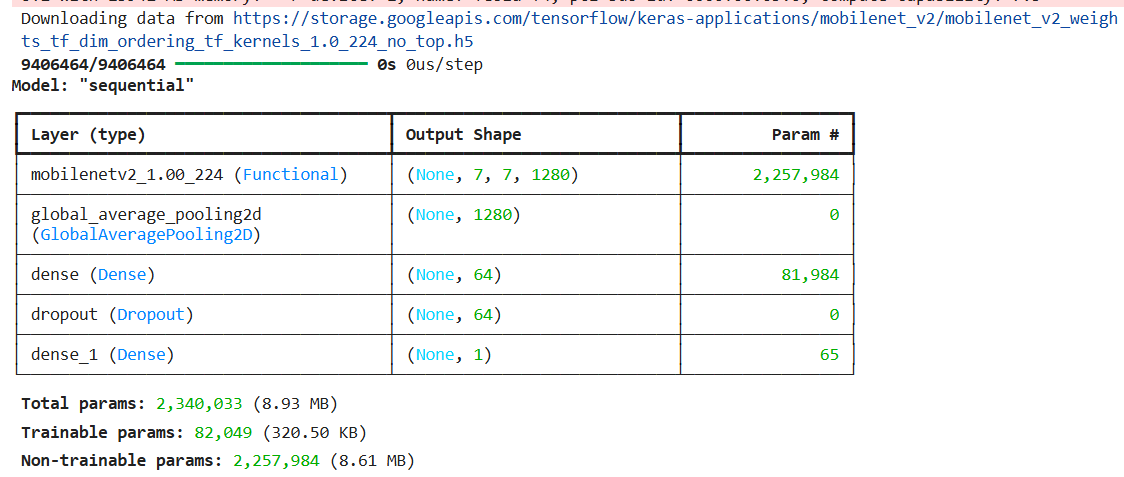




Hình 3.9 kết quả huấn luyện mô hình Resnet50



\*Huấn luyện mô hình để phân loại ảnh đầu vào có phải ảnh lá hay không?



Hình 3.10 Quy trình và kết quả huấn luyện mô hình nhận diện ảnh lá

# KẾT LUẬN

Trong đề tài “Xây dựng hệ thống nhận diện bệnh lá cây”, nhóm đã ứng dụng hiệu quả các kiến thức về thị giác máy tính (Computer Vision) và học sâu (Deep Learning) để xây dựng một hệ thống có khả năng phân tích hình ảnh lá cây và tự động nhận diện các loại bệnh thường gặp.

Hệ thống được thiết kế theo kiến trúc pipeline gồm nhiều giai đoạn: từ tiền xử lý ảnh, tăng cường dữ liệu (data augmentation), trích xuất đặc trưng cho đến huấn luyện mô hình phân loại. Trong quá trình thực nghiệm, nhóm đã thử nghiệm các kiến trúc mạng nơ-ron tích chập (CNN) khác nhau như MobileNet, ResNet, đồng thời so sánh hiệu quả giữa các phương pháp tăng cường dữ liệu để cải thiện độ chính xác và khả năng khái quát hóa của mô hình.

Kết quả thu được cho thấy mô hình đạt độ chính xác cao và ổn định trong việc nhận diện một số loại bệnh lá phổ biến, đặc biệt là khi ảnh đầu vào có chất lượng tốt và điều kiện ánh sáng ổn định. Hệ thống cũng đã được tích hợp vào một giao diện đơn giản, thân thiện với người dùng, cho phép tải ảnh lên, hiển thị kết quả phân loại và thông tin bệnh tương ứng một cách trực quan.

Đề tài không chỉ giúp nhóm củng cố và mở rộng kiến thức về AI trong nông nghiệp, mà còn rèn luyện kỹ năng xử lý ảnh, thiết kế mô hình học sâu, xây dựng giao diện người dùng và triển khai hệ thống thực tiễn.

**Hạn chế của nghiên cứu**

Mặc dù hệ thống đã đạt được một số kết quả tích cực, đề tài vẫn còn tồn tại một số điểm hạn chế như sau:

* Giới hạn về dữ liệu huấn luyện: Bộ dữ liệu ảnh được sử dụng chủ yếu thu thập từ nguồn mở hoặc được gán nhãn thủ công, nên số lượng và tính đa dạng còn hạn chế. Hệ thống chưa được kiểm chứng với các giống cây trồng khác nhau, điều kiện ánh sáng, góc chụp và môi trường thực tế ngoài đồng ruộng.
* Khả năng phân biệt giữa các bệnh tương tự còn yếu: Một số loại bệnh có triệu chứng hình ảnh gần giống nhau dẫn đến nhầm lẫn trong phân loại. Điều này đòi hỏi mô hình cần thêm dữ liệu chất lượng và kỹ thuật tăng cường chuyên biệt hơn.
* Thiếu khai thác đặc trưng nâng cao: Mô hình chủ yếu dựa vào CNN cơ bản mà chưa áp dụng các kỹ thuật thị giác máy tính hiện đại hơn như attention mechanism, multi-scale feature extraction hoặc mô hình thị giác tiên tiến như Vision Transformers (ViT).
* Chưa kiểm thử trong môi trường thực tế: Hệ thống mới chỉ được thử nghiệm trong môi trường mô phỏng. Chưa có đánh giá hiệu suất khi triển khai trên thiết bị di động, trong điều kiện mạng yếu hoặc khi người dùng sử dụng camera chất lượng thấp.

**Hướng phát triển**

Để nâng cao hiệu quả và khả năng ứng dụng thực tế của hệ thống, nhóm đề xuất một số hướng phát triển trong tương lai như sau:

* Mở rộng và làm giàu tập dữ liệu: Thu thập thêm ảnh lá cây bị bệnh từ nhiều nguồn khác nhau, bao gồm dữ liệu thực tế ngoài hiện trường, nhiều giống cây trồng và nhiều loại bệnh hơn. Đồng thời áp dụng kỹ thuật gán nhãn bán tự động để giảm chi phí xử lý dữ liệu.
* Ứng dụng các mô hình thị giác tiên tiến: Tích hợp các kiến trúc mô hình mạnh hơn như EfficientNet, MobileNetV3 hoặc Vision Transformers để cải thiện khả năng trích xuất đặc trưng và nhận diện chính xác hơn, đặc biệt trong trường hợp hình ảnh nhiễu hoặc bệnh có biểu hiện tương đồng.
* Tích hợp hệ thống trên thiết bị di động: Phát triển ứng dụng di động (Android/iOS) để người dùng có thể chụp ảnh lá cây trực tiếp bằng điện thoại và nhận kết quả phân tích tức thì, giúp tăng khả năng tiếp cận và ứng dụng trong thực tế sản xuất nông nghiệp.
* Bổ sung chức năng chẩn đoán và gợi ý biện pháp xử lý: Hệ thống không chỉ dừng lại ở việc phân loại bệnh mà có thể mở rộng thêm phần kiến thức hỗ trợ, đưa ra nguyên nhân, triệu chứng và khuyến nghị biện pháp xử lý phù hợp.
* Kết hợp dữ liệu đa nguồn: Nâng cao độ chính xác bằng cách kết hợp dữ liệu ảnh với các yếu tố khác như điều kiện thời tiết, đất đai, giai đoạn sinh trưởng của cây… để mô hình đưa ra đánh giá toàn diện hơn.
* Triển khai thử nghiệm thực tế: Hợp tác với các hợp tác xã nông nghiệp, doanh nghiệp hoặc cơ quan quản lý nông nghiệp để triển khai thí điểm hệ thống trong môi trường canh tác thực tế và đánh giá hiệu quả sử dụng.

# TÀI LIỆU THAM KHẢO

[1] John Smith, 2023, *Explanation of convolutional neural networks* (CNNs) and their applications in computer vision, *Encord Tech Review*

[2] Emily Johnson, 2022, Overview of supervised machine learning and how algorithms learn from labeled data, *FPT Digital Insights*

[3] David Lee, 2021, Understanding CNN architecture with convolution and pooling layers to reduce computational cost, *OpenCV Journal*

[4] Sophia Martinez, 2020, Analysis of the K-Nearest Neighbors algorithm and its application in data classification, *GeeksforGeeks Science*

[5] Michael Brown, 2024, Deep learning-based image classification methods and their impact on model accuracy, *Artificial Intelligence Review (Springer)*

[6] Transfer learning with fine‑tuned deep CNN ResNet50 model for COVID‑19 classification, 2022, *ScienceDirect*

[7] Overview of VGG16, ResNet50, Xception and MobileNet Neural Networks, Tannaz Mostafid, Dec 11 2023, *Medium*

[8] A comparative analysis of eleven neural networks architectures…, 2021, *PMC/Nature*

[9] Comparison of MobileNet and ResNet CNN Architectures in CNN‑Based Skin Cancer Classifier Model, 2020, *ResearchGate*

[10] Machine Learning for Healthcare Applications, 2021, *Wiley*

[11] PlantDoc: A Dataset for Visual Plant Disease Detection, Singh et al., 2019

[12] The Plant Pathology 2020 challenge dataset to classify foliar disease of apples, Thapa et al., 2020

[13] Unsupervised and Supervised Feature Extraction Methods … based on mixtures of factor analyzers, Zhao et al., 2020, *Remote Sensing (MDPI)*

[14] CNN features are also great at unsupervised classification, Guérin et al., 2017

[15] Unsupervised Deep Feature Extraction for Remote Sensing Image Classification, Romero et al., 2015, *IEEE TGRS*

[16] Medical Image Segmentation Using Deep Learning: A Survey, Wang et al., 2020

[17] An overview of unsupervised and generalizable image segmentation, 2024

[18] A review of feature extraction techniques for image analysis, Goel et al., 2017