ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN
Website: http://dut.udn.vn/khoacntt, E-mail: cntt@dut.udn.vn

BÁO CÁO KẾT THÚC HỌC PHẦN
HỌC MÁY VÀ ỨNG DỤNG

ĐỀ TÀI:

FLOWER CLASSIFICATION

GIẢNG VIÊN HƯỚNG DẪN: PGS.TS. Phạm Công Thắng

SINH VIÊN	MSSV	LỚP HỌC PHẦN
Trương Quang Hải	102220013	22N15A
Nguyễn Khả Hào	102220014	22N15A
Trần Phi Hùng	102220020	22N15A

Đà Nẵng, 2025-12-

TÓM TẮT ĐỒ ÁN

Trong đề tài này, chúng tôi tập trung nghiên cứu và xây dựng hệ thống phân loại hình ảnh các loài hoa dựa trên phương pháp học sâu (Deep Learning). Bài toán phân loại hoa là một dạng phân loại ảnh đa lớp (multi-class classification), yêu cầu mô hình có khả năng nhận diện và trích xuất đặc trưng hình ảnh chính xác trong điều kiện dữ liệu đa dạng về hình dạng, màu sắc và góc chụp.

Chúng tôi đã tiến hành thử nghiệm và so sánh hiệu quả của nhiều kiến trúc mạng CNN hiện đại, bao gồm EfficientNet, ResNet và MobileNet, nhằm đánh giá hiệu suất và khả năng tổng quát hóa của từng mô hình. Đồng thời, đề tài ứng dụng các kỹ thuật tối ưu như Data Augmentation, Class Weight và Focal Loss để cải thiện chất lượng dự đoán đối với tập dữ liệu mất cân bằng và giảm hiện tượng overfitting.

Kết quả thực nghiệm cho thấy mô hình EfficientNet và ResNet đạt độ chính xác cao hơn nhờ khả năng trích chọn đặc trưng mạnh mẽ, trong khi MobileNet có ưu điểm về tốc độ suy luận và kích thước nhẹ, phù hợp triển khai trên thiết bị di động. Độ chính xác cao nhất trên tập kiểm thử đạt khoảng 87–90%, chứng minh hiệu quả của phương pháp đề xuất.

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên	Các nhiệm vụ	Đánh giá
Trương Quang Hải	Thu thập ảnh từ các loài	Đã hoàn thành
	Lọc thủ công dữ liệu	Đã hoàn thành
	Xây dựng mô hình ResNet50	Đã hoàn thành
	Huấn luyện và đánh giá mô hình	Đã hoàn thành
	Xây dựng ứng dụng Web	Đã hoàn thành
	Báo cáo và thuyết trình	Đã hoàn thành
Nguyễn Khả Hào	Thu thập ảnh từ các loài	Đã hoàn thành
	Lọc thủ công dữ liệu	Đã hoàn thành
	Xây dựng mô hình ResNet50	Đã hoàn thành
	Huấn luyện và đánh giá mô hình	Đã hoàn thành
	Xây dựng ứng dụng Web	Đã hoàn thành
	Báo cáo và thuyết trình	Đã hoàn thành
Trần Phi Hùng	Thu thập tên các loài	Đã hoàn thành
	Lọc thủ công dữ liệu	Đã hoàn thành
	Xây dựng mô hình ResNet50	Đã hoàn thành
	Huấn luyện và đánh giá mô hình	Đã hoàn thành
	Xây dựng ứng dụng Web	Đã hoàn thành
	Báo cáo và thuyết trình	Đã hoàn thành

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU 1

1.1. Tổng quan 1
1.2. Mục tiêu nghiên cứu 1
1.3. Đối tượng và phạm vi nghiên cứu 2
1.3.1. Đối tượng nghiên cứu 2
1.3.2. Phạm vi nghiên cứu 2
1.4. Phương pháp nghiên cứu 3

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 4

2.1. Giới thiệu chung 4
2.2. Thị giác máy tính 5
2.2.1. Khái niệm 5
2.2.2. Các nhiệm vụ chính của Computer Vision 5
2.2.3. Cách hoạt động của Computer Vision 6
2.2.4. Các ứng dụng của Computer Vision 7
2.3. Học máy (Machine Learning) và học sâu (Deep Learning) 9
2.3.1. Học máy là gì? 9
2.3.2. Các phương pháp học máy phổ biến 10
2.3.3. Quy trình hoạt động cơ bản 10
2.3.4. Tổng quan học sâu 11
2.3.5. Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN) 11
2.4. Mô hình EfficientNetB0 17
2.4.1. Giới thiệu EfficientNet 17
2.4.2. Ý tưởng chính của EfficientNet 17
2.4.3. Kiến trúc EfficientNetB0 19
2.5. Mô hình Resnet50 19
2.5.1. Giới thiệu mô hình 19
2.5.2. Vanishing Gradient 20
2.5.3. Residual Block 21
2.5.4. Khả năng mở rộng mạng sâu với ResNet 22
2.5.5. Kiến trúc mạng ResNet50 23
2.6. Mô hình MobileNet 24
2.6.1. Giới thiệu mô hình 24
2.6.2. Standard Convolutions 25
2.6.3. Kiến trúc MobileNet 26
2.6.4. Mô hình cải tiến MobileNetV2 và MobileNetV3 28
2.7. Tăng cường dữ liệu 31
2.7.1. Điều chỉnh ảnh 31
2.7.2. Phương pháp MixUp 31
2.8. Các kĩ thuật tối ưu hóa huấn luyện 32
2.8.1. Class Weight 32
2.8.2. Focal Loss 33
2.9. Các chỉ số đánh giá mô hình 34

2.9.1. Accuracy (Đô chính xác) ... 34
2.9.2. Loss (Hàm mất mát) ... 34
2.9.3. Classification Report ... 35

CHƯƠNG 3: DỮ LIỆU HUẤN LUYỆN VÀ TIỀN XỬ LÝ ... 37

3.1. Nguồn dữ liệu ... 37
3.2. Quá trình thu thập dữ liệu ... 38
3.2.1. Thu thập tên khoa học ... 38
3.2.2. Thu thập ảnh theo loài ... 39
3.3. Tiền xử lý ảnh làm bộ dữ liệu ... 40
3.3.1. Chuẩn hóa kích thước ảnh (Image Resizing) ... 40
3.4. Phân chia tập dữ liệu ... 42
3.5. Trực quan hóa dữ liệu ... 43
3.5.1. Biểu đồ phân phối ảnh theo loài hoa ... 43
3.5.2. Trực quan hóa so sánh các tập dữ liệu ... 44

CHƯƠNG 4: XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH ... 47

4.1. Các kỹ thuật huấn luyện mô hình ... 47
4.1.1. Tăng cường dữ liệu ... 47
4.1.2. Class Weights ... 48
4.1.3. Focal loss ... 48
4.1.4. ReduceLROnPlateau ... 49
4.2. Mô hình EfficientNetB0 ... 50
4.2.1. Kiến trúc mô hình ... 50
4.2.2. Siêu tham số ... 50
4.2.3. Quá trình huấn luyện ... 51
4.3. Mô hình Resnet ... 51
4.4. Mô hình MobileNet ... 54
4.4.1. Kiến trúc mô hình ... 54
4.4.2. Siêu tham số ... 54
4.4.3. Quá trình huấn luyện ... 55

CHƯƠNG 5: ĐÁNH GIÁ VÀ SO SÁNH MÔ HÌNH ... 57

5.1. Kết quả trên tập test ... 57
5.1.1. Mô hình EfficientNetB0 ... 57
5.1.2. Mô hình Resnet ... 58
5.1.3. Mô hình MobileNet ... 60
5.2. So sánh mô hình ... 61

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ... 62

6.1. Kết luận ... 62
6.2. Hướng phát triển ... 62

DANH MỤC HÌNH ẢNH, BẢNG BIỂU

Figure 2-1: Ảnh minh họa thị giác máy tính 5
Figure 2-2: Nhận diện khuôn mặt 7
Figure 2-3: Tìm kiếm hình ảnh 7
Figure 2-4: Xe tự lái 8
Figure 2-5: Y tế 8
Figure 2-6: Ứng dụng giao thông thông minh 9
Figure 2-7: Học máy 9
Figure 2-8: Các phương pháp học máy phổ biến 10
Figure 2-9: Mạng nơ-ron nhân tạo 12
Figure 2-10: Nguyên lý hoạt động của mạng ANN 13
Figure 2-11: Minh họa compound scaling 18
Figure 2-12: Kiến trúc EfficientNetB0 19
Figure 2-13: Minh họa huấn luyện mạng CNN 20
Figure 2-14: Vanishing Gradient 21
Figure 2-15: Residual Block 21
Figure 2-16: Các mô hình CNN 22
Figure 2-17: Kiến trúc mạng ResNet50 23
Figure 2-18: Standard Convolutions 25
Figure 2-19: Depthwise Convolutions 26
Figure 2-20: Pointwise Convolution 26
Figure 2-21: So sánh stardard CNN và MobileNet 27
Figure 2-22: So sánh MobileNetV1 và V2 28
Figure 2-23: Kiến trúc MobileNetV2 29
Figure 2-24: Kiến trúc MobileNetV3 29
Figure 2-25: Kiến trúc tinh giản Layer 30
Figure 2-26: Kĩ thuật MixUp 32
Figure 3-1: Kiểm tra tên khoa học 38
Figure 3-2: File CSV lưu tên khoa học 39
Figure 3-3: API inaturalist.org 39
Figure 3-4: Đa luồng thu thập ảnh 39
Figure 3-5: Tổ chức thư mục sau khi thu thập 40
Figure 3-6: Kích thước ảnh 40

Figure 3-7: Resize ảnh 41
Figure 3-8: Ảnh sau khi lọc thủ công.....42
Figure 3-9: Cấu trúc thư mục sau khi chia 43
Figure 3-10: Phân phối dữ liệu theo loài.....43
Figure 3-11: Phân phối dữ liệu theo loài trên tập train 44
Figure 3-12: Số lượng ảnh từng tập dữ liệu 45
Figure 3-13: Tỷ lệ phân chia thực tế 46
Figure 4-1: Phương pháp ImageDataGenerator 47
Figure 4-2: Phương pháp MixUp 47
Figure 4-3: Phương pháp Class Weight 48
Figure 4-4: Phương pháp Focal Loss 48
Figure 4-5: Phương pháp ReduceLROnPlateau.....49
Figure 4-6: Build model EfficientNetB0.....50
Figure 4-7: Quá trình huấn luyện EfficientNetB0.....51
Figure 4-8: Build model ResNet50 52
Figure 4-9: Quá trình huấn luyện ResNet50 53
Figure 4-10: Build model MobileNet.....54
Figure 4-11: Quá trình huấn luyện MobileNet không dùng phương pháp.....55
Figure 4-12: Quá trình huấn luyện MobileNet có dùng phương pháp.....56
Figure 5-1: Kết quả trên tập Test (EfficientNetB0) 57
Figure 5-2: Phân phối Accuracy từng loài trên tập Test (EfficientNetB0) 57
Figure 5-3: Confusion Matrix (EfficientNetB0) 58
Figure 5-4: Kết quả trên tập Test (ResNet50).....58
Figure 5-5: Top 5 class tốt nhất và tệ nhất (Resnet 50) 59
Figure 5-6: Confusion Matrix (ResNet50).....59
Figure 5-7: Kết quả trên tập Test (MobileNet) (1).....60
Figure 5-8: Phân phối Accuracy từng loài trên tập Test (MobileNet) (1).....60
Figure 5-9: Kết quả trên tập Test (MobileNet) (2).....60
Figure 5-10: Phân phối Accuracy từng loài trên tập Test (MobileNet) (2).....60

Table 2-1: Điều chỉnh ảnh.....31
Table 3-1: Phân chia tập dữ liệu.....42
Table 5-1: Bảng so sánh 2 phương pháp MobileNet.....61
Table 5-2: So sánh mô hình 61

CHƯƠNG 1: GIỚI THIỆU

1.1. Tổng quan

Trong thời đại chuyển đổi số và công nghệ 4.0, dữ liệu hình ảnh đang phát triển với tốc độ rất nhanh, trong đó hình ảnh về thiên nhiên, đặc biệt là hoa, chiếm một phần lớn. Việc tự động nhận diện và phân loại các loài hoa từ hình ảnh không chỉ hỗ trợ nghiên cứu khoa học, bảo tồn sinh học, nông nghiệp thông minh, mà còn phục vụ du lịch, giáo dục và các ứng dụng trực tuyến khác.

Bài toán **Flower Classification** đòi hỏi hệ thống máy tính phải nhận diện các đặc trưng hình dạng, màu sắc, kết cấu cánh hoa và nhị hoa, sau đó gán nhãn chính xác cho từng loài. Đây là sự kết hợp giữa Thị giác máy tính (Computer Vision)**,** giúp máy "nhìn" và trích xuất đặc trưng hình ảnh, và các phương pháp học máy (Machine Learning), giúp máy hiểu và phân loại các mẫu hoa khác nhau.

Từ hình ảnh một bông hoa có màu đỏ, cánh tròn và nhị vàng, hệ thống có thể nhận diện và phân loại thành hoa hồng đỏ. Khi dữ liệu hình ảnh về hoa ngày càng đa dạng, việc áp dụng AI giúp tự động hóa quá trình phân loại, tiết kiệm thời gian và nâng cao độ chính xác so với phương pháp thủ công.

1.2. Mục tiêu nghiên cứu

Mục tiêu chính của nghiên cứu là phát triển một mô hình tự động phân loại các loài hoa từ hình ảnh với độ chính xác cao, nhằm hỗ trợ việc nhận diện và quản lý dữ liệu hoa một cách hiệu quả. Cụ thể, nghiên cứu hướng tới:

- Trích xuất đặc trưng hình ảnh: Xây dựng phương pháp xử lý ảnh để nhận diện các đặc trưng quan trọng của hoa như hình dạng, màu sắc, kết cấu cánh hoa và nhị hoa.
- Phân loại chính xác các loài hoa: Áp dụng các thuật toán học máy hoặc học sâu (Machine Learning / Deep Learning) để phân loại hình ảnh hoa thành các loài tương ứng dựa trên đặc trưng đã trích xuất.
- Tự động hóa và tiết kiệm thời gian: Giảm sự phụ thuộc vào phương pháp phân loại thủ công, giúp xử lý khối lượng lớn dữ liệu hình ảnh hoa nhanh chóng và hiệu quả.

- Đánh giá hiệu năng mô hình: So sánh và đánh giá độ chính xác, tốc độ và khả năng tổng quát hóa của mô hình trên các bộ dữ liệu hoa khác nhau, từ đó đề xuất giải pháp cải thiện hiệu suất.

Thông qua các mục tiêu này, mô hình không chỉ giúp nhận diện chính xác từng loài hoa mà còn mở ra cơ hội ứng dụng trong nghiên cứu sinh học, nông nghiệp thông minh, giáo dục, và các ứng dụng trực tuyến liên quan đến thiên nhiên.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là các hình ảnh về hoa thuộc nhiều loài khác nhau. Trong đó, mô hình hướng tới việc nhận diện và phân loại dựa trên các đặc trưng hình ảnh quan trọng như:

- Hình dạng và kết cấu cánh hoa
- Màu sắc chủ đạo
- Nhị hoa và các chi tiết đặc trưng khác

Mục tiêu là xây dựng một mô hình AI có khả năng phân biệt và gán nhãn chính xác các loài hoa từ những đặc trưng này, bất kể hình ảnh có điều kiện ánh sáng hay góc chụp khác nhau.

1.3.2. Phạm vi nghiên cứu

Đề tài tập trung vào:

- Loại dữ liệu: Chỉ sử dụng hình ảnh hoa (không bao gồm video hay dữ liệu 3D).
- Số lượng loài hoa: Giới hạn nghiên cứu trên một bộ dữ liệu gồm các loài hoa phổ biến hoặc có sẵn trong bộ dữ liệu nghiên cứu (ví dụ: 10–50 loài).
- Phương pháp: Sử dụng các thuật toán học máy và học sâu để trích xuất đặc trưng và phân loại, không nghiên cứu về lai tạo hoa hay sinh học phân tử.
- Ứng dụng: Ứng dụng mô hình chủ yếu trong việc tự động nhận diện và phân loại hoa từ hình ảnh, phục vụ nghiên cứu sinh học, giáo dục, nông nghiệp thông minh và các ứng dụng trực tuyến liên quan.

1.4. Phương pháp nghiên cứu

Để đạt được mục tiêu phân loại các loài hoa từ hình ảnh, nghiên cứu sẽ áp dụng các phương pháp sau:

- **Thu thập và chuẩn bị dữ liệu**
  - + Thu thập hình ảnh hoa bằng phương pháp cào ảnh từ các trang web thiên nhiên (như inaturalist, …).
  - + Tiền xử lý hình ảnh: cắt, chuẩn hóa kích thước, cân bằng màu sắc, loại bỏ nhiễu để đảm bảo chất lượng dữ liệu đầu vào.
  - + Gán nhãn hình ảnh theo từng loài hoa để tạo tập dữ liệu huấn luyện, kiểm tra và đánh giá.
- **Trích xuất đặc trưng hình ảnh**
  - + Sử dụng các kỹ thuật Thị giác máy tính (Computer Vision) để nhận diện đặc trưng của hoa: hình dạng cánh, màu sắc chủ đạo, kết cấu bề mặt, nhị hoa.
  - + Áp dụng các phương pháp học sâu như Convolutional Neural Networks (CNNs) để tự động học đặc trưng từ hình ảnh mà không cần định nghĩa thủ công.
- **Xây dựng mô hình phân loại**
  - + Huấn luyện các mô hình học máy hoặc học sâu (CNN, ResNet, EfficientNet, v.v.) trên tập dữ liệu đã chuẩn bị.
  - + Tối ưu hóa tham số mô hình để đạt độ chính xác cao nhất, đồng thời tránh hiện tượng quá khớp (overfitting).
- **Đánh giá và kiểm tra mô hình**
  - + Sử dụng các chỉ số đánh giá tiêu chuẩn: độ chính xác (accuracy), độ nhạy (precision), độ bao quát (recall), F1-score.
  - + Kiểm tra khả năng tổng quát hóa của mô hình trên các hình ảnh hoa chưa xuất hiện trong tập huấn luyện.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu chung

Chương này trình bày các khái niệm, lý thuyết và các phương pháp nền tảng phục vụ cho việc phân loại các loài hoa từ hình ảnh. Mục tiêu của chương là cung cấp kiến thức cơ sở để hiểu rõ nguyên lý hoạt động của các mô hình học máy và học sâu, cũng như các kỹ thuật xử lý và trích xuất đặc trưng hình ảnh, giúp mô hình phân loại hoa đạt độ chính xác cao.

Nội dung chính của chương bao gồm:

- **Thị giác máy tính (Computer Vision):** Giới thiệu các kỹ thuật nhận diện và trích xuất đặc trưng từ hình ảnh, bao gồm hình dạng, màu sắc, kết cấu cánh hoa và nhị hoa.
- **Học máy và học sâu (Machine Learning & Deep Learning):** Trình bày các thuật toán phân loại, từ các mô hình truyền thống đến mạng nơ-ron tích chập (CNN), cùng các phương pháp huấn luyện và tối ưu hóa mô hình.
- **Phương pháp kĩ thuật huấn luyện:** Điều chỉnh siêu tham số (learning rate, batch size, số lớp, …), các kĩ thuật đối với dữ liệu không cân bằng.
- **Các mô hình áp dụng cho bài toán:** Các mô hình học sâu trích xuất đặc trưng hình ảnh, khả năng xử lý nhiều lớp và học đặc trưng phức tạp.
- **Tăng cường dữ liệu (Data Augmentation):** Giới thiệu các kỹ thuật tăng cường dữ liệu như xoay, lật, phóng to/thu nhỏ, thay đổi độ sáng hoặc màu sắc, giúp mở rộng tập dữ liệu, cải thiện khả năng tổng quát hóa của mô hình.
- **Các phương pháp đánh giá mô hình:** Giới thiệu các chỉ số đánh giá như độ chính xác, độ nhạy, F1-score, giúp đánh giá hiệu quả và so sánh các mô hình phân loại.

Chương cơ sở lý thuyết này tạo nền tảng vững chắc cho việc phát triển mô hình phân loại hoa trong các chương tiếp theo, đồng thời cung cấp các kiến thức cần thiết để lựa chọn phương pháp và thuật toán phù hợp với đề tài.

2.2. Thị giác máy tính

2.2.1. Khái niệm

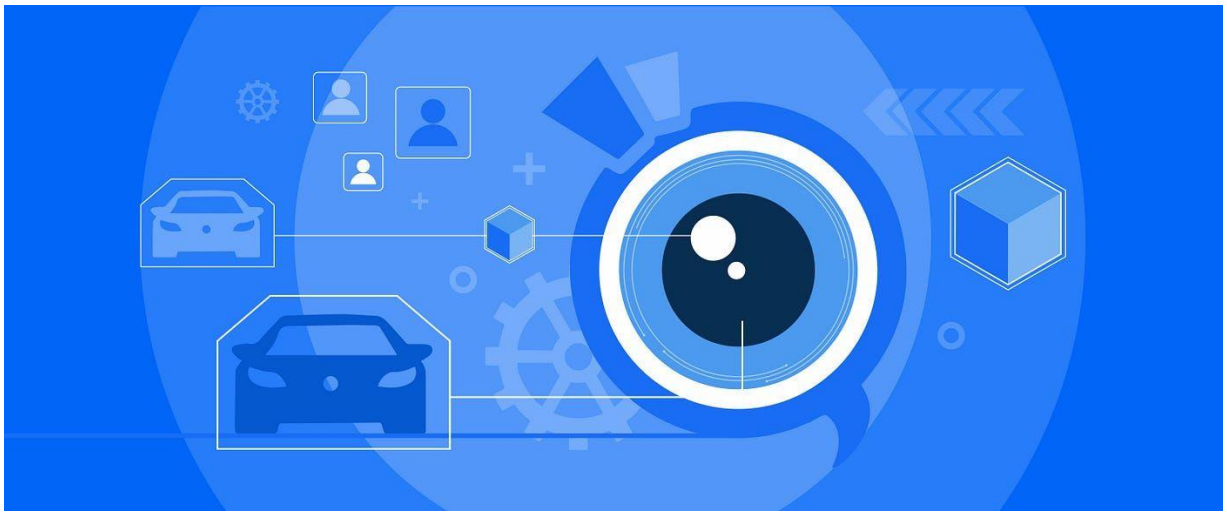


Figure 2-1: Ảnh minh họa thị giác máy tính

Computer Vision (Thị giác máy tính) là một lĩnh vực khoa học máy tính tập trung vào việc cho phép máy tính "nhìn" và "hiểu" nội dung của hình ảnh hoặc video. Nói cách khác, nó giúp máy tính có khả năng trích xuất thông tin hữu ích từ dữ liệu hình ảnh.

Computer Vision là một trong những công nghệ liên quan đến Trí tuệ nhân tạo (AI) được sử dụng nhiều trong đời sống hàng ngày. Các nhiệm vụ mà Computer Vision có thể thực hiện được bao gồm: nhận dạng hình ảnh, phát hiện đối tượng, ...

2.2.2. Các nhiệm vụ chính của Computer Vision

- **Nhận dạng hình ảnh (Image Recognition):** Đây là ứng dụng phổ biến nhất, hệ thống xác định một đối tượng, người hoặc hành động cụ thể trong hình ảnh.
- **Phát hiện đối tượng (Object Detection):** Nhận dạng nhiều đối tượng trong một hình ảnh và xác định vị trí của chúng bằng hộp giới hạn. Được sử dụng rộng rãi trong các ứng dụng AI như xe tự lái, nơi cần phải nhận dạng tất cả các đối tượng có liên quan xung quanh xe.
- **Phân đoạn hình ảnh (Image Segmentation):** Phân chia một hình ảnh thành nhiều phân đoạn để đơn giản hóa hoặc thay đổi cách biểu diễn hình ảnh thành thứ gì đó có ý nghĩa hơn và dễ phân tích hơn. Nó thường được sử dụng trong hình ảnh y tế.

- **Nhận dạng khuôn mặt (Facial Recognition):** Hệ thống nhận dạng hoặc xác minh một người từ hình ảnh kỹ thuật số hoặc khung hình video.
- **Phân tích chuyển động (Motion Analysis):** Nhận dạng quỹ đạo của các vật thể chuyển động trong video, thường được sử dụng trong an ninh, giám sát và phân tích thể thao.
- **Thị giác máy (Machine Vision):** Kết hợp thị giác máy tính với robot để xử lý dữ liệu trực quan và kiểm soát chuyển động của phần cứng trong các ứng dụng như dây chuyền lắp ráp tự động trong nhà máy.

2.2.3. Cách hoạt động của Computer Vision

Thị giác máy tính là lĩnh vực tập trung vào nhận dạng mẫu (pattern recognition), với mục tiêu "dạy" máy tính hiểu dữ liệu, thông qua việc cung cấp cho nó hàng nghìn hoặc hàng triệu hình ảnh đã được gắn nhãn, rồi áp dụng các thuật toán hoặc kỹ thuật khác nhau để máy tính học và nhận diện đúng (vẫn có xác xuất nhận diện sai) khi gặp lại các hình ảnh tương tự trong tương lai (hình ảnh người, xe hơi, con vật, ...).

Một dự án Computer Vision sẽ bao gồm các bước cơ bản sau đây:

- **Thu thập dữ liệu hình ảnh:** Thu thập một lượng lớn hình ảnh hoặc video. Các hình ảnh này có thể đã được gắn nhãn hoặc không gắn nhãn (supervised hoặc unsupervised). Gắn nhãn là việc xác định các đối tượng trong hình ảnh (ví dụ: "mèo", "xe hơi", v.v.).
- **Xử lý dữ liệu hình ảnh**: Dữ liệu hình ảnh sau khi thu thập sẽ được xử lý để chuẩn bị cho việc phân tích. Các thuật toán xử lý hình ảnh thực hiện các bước như điều chỉnh độ sáng, cắt xén (cropping), thay đổi kích thước và loại bỏ nhiễu.
- **Trích xuất đặc trưng (Feature Extraction):** Các thuật toán Computer Vision sẽ trích xuất các đặc trưng quan trọng từ hình ảnh, như các cạnh, góc, hình dạng, ... Một trong các phương pháp nổi tiếng là Convolutional Neural Networks (CNN).
- **Huấn luyện mô hình (Model Training):** Các mô hình học máy (VD: CNN) sẽ được huấn luyện trên tập dữ liệu hình ảnh đã qua xử lý. Mô hình sẽ học cách

phân loại hoặc nhận dạng các đối tượng trong ảnh dựa trên các đặc trưng đã trích xuất.

– **Dự đoán và phân loại:** Dựa vào các đặc trưng đã học, mô hình sẽ đưa ra dự đoán về các đối tượng xuất hiện trong ảnh mới (chưa được gắn nhãn). Quá trình này diễn ra tự động mà không cần sự can thiệp của con người.

2.2.4. Các ứng dụng của Computer Vision

– **Nhận diện khuôn mặt (Face Recognition):** Nhận diện khuôn mặt được sử dụng trong nhiều lĩnh vực như mở khóa điện thoại, bảo mật hệ thống, và kiểm tra danh tính.

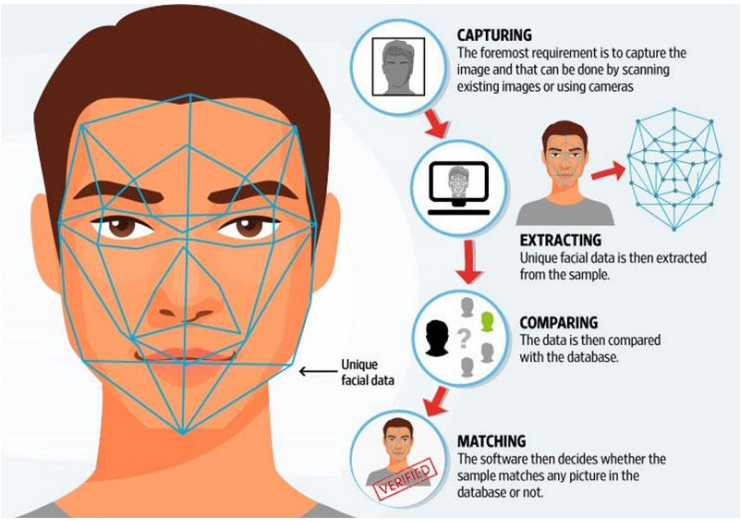


Figure 2-2: Nhận diện khuôn mặt

– **Tìm kiếm hình ảnh (Image Search):** Các công cụ tìm kiếm hình ảnh cho phép người dùng tìm kiếm sản phẩm hoặc nội dung tương tự chỉ bằng cách tải lên một hình ảnh.

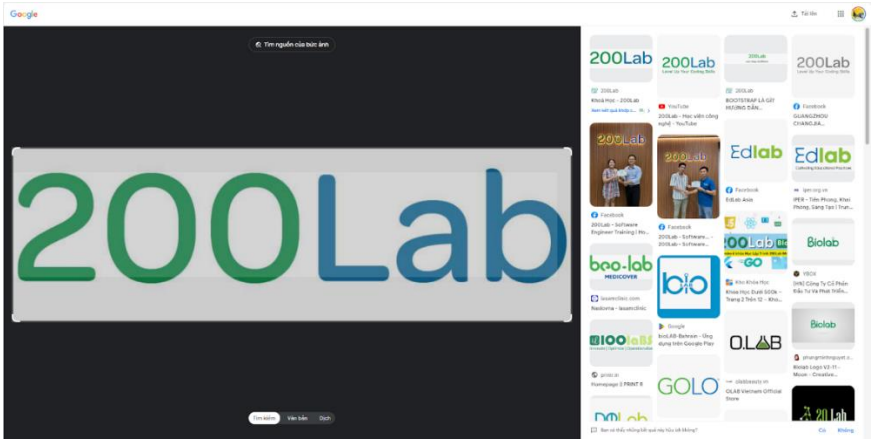


Figure 2-3: Tìm kiếm hình ảnh

– **Xe tự lái (Autonomous Vehicles):** Thị giác máy tính giúp xe tự hành "nhìn" và phân tích môi trường xung quanh như làn đường, biển báo, các phương tiện khác và người đi bộ để đưa ra quyết định lái xe an toàn.

Figure 2-4: Xe tự lái

– **Y tế (Healthcare):** Trong y tế, thị giác máy tính được sử dụng để phân tích hình ảnh chụp từ các thiết bị y khoa như X-quang, MRI để hỗ trợ bác sĩ trong việc chẩn đoán bệnh.

Figure 2-5: Y tế

– **Ứng dụng giao thông thông minh (Smart Traffic Applications):** Thị giác máy tính được áp dụng để giám sát giao thông, nhận diện biển số xe, phát hiện vi phạm giao thông, và quản lý tín hiệu giao thông tự động.

Figure 2-6: Ứng dụng giao thông thông minh

2.3. Học máy (Machine Learning) và học sâu (Deep Learning)

2.3.1. Học máy là gì?

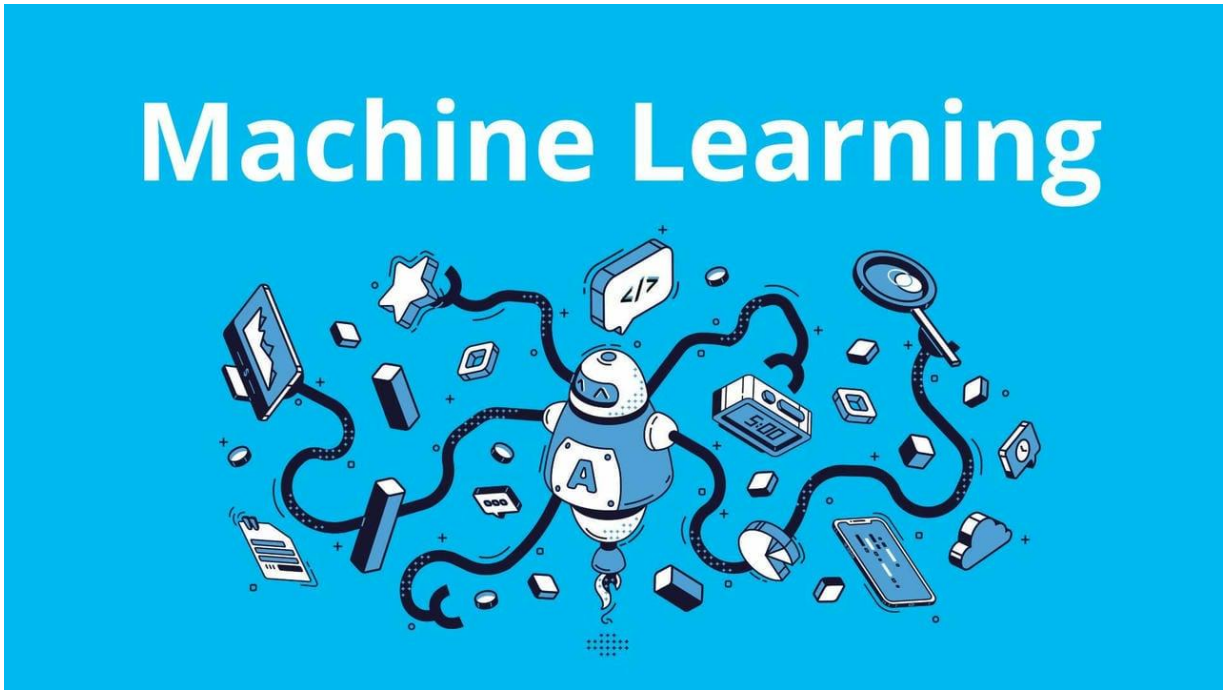


Figure 2-7: Học máy

Machine Learning (học máy), được viết tắt là ML, là một lĩnh vực thuộc trí tuệ nhân tạo (AI), tập trung vào phát triển các thuật toán cho phép máy tính tự động cải thiện

và tăng độ chính xác thông qua kinh nghiệm và dựa trên dữ liệu. Nói một cách đơn giản, ML cho phép máy tính học từ dữ liệu và đưa ra quyết định hoặc dự đoán mà không cần được lập trình rõ ràng cho từng tác vụ.

2.3.2. Các phương pháp học máy phổ biến

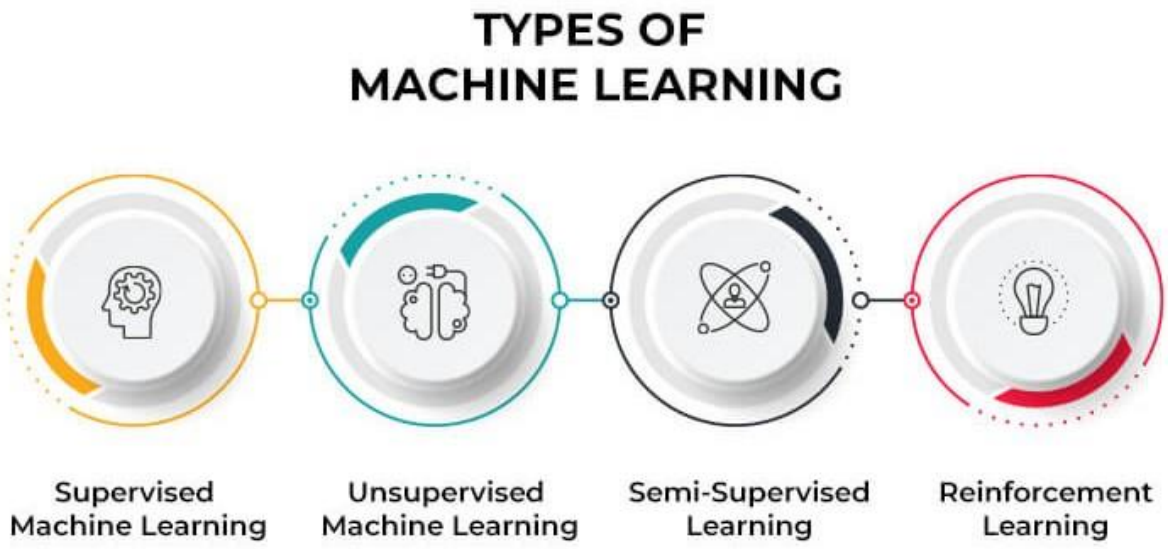


Figure 2-8: Các phương pháp học máy phổ biến

Học máy có thể chia thành các phương pháp phổ biến:

- **Học có giám sát (Supervised Learning):** Sử dụng dữ liệu đã gắn nhãn để huấn luyện, dự đoán hoặc phân loại dữ liệu mới.
- **Học không giám sát (Unsupervised Learning):** Dữ liệu không có nhãn, thuật toán tự tìm các nhóm, mẫu, cấu trúc ẩn.
- **Học bán giám sát (Semi-supervised Learning):** Kết hợp dữ liệu có nhãn và không nhãn để nâng cao hiệu suất khi dữ liệu có nhãn hạn chế.
- **Học tăng cường (Reinforcement Learning):** Hệ thống học cách hành động trong môi trường dựa vào phần thưởng hoặc hình phạt từ môi trường.

2.3.3. Quy trình hoạt động cơ bản

Quy trình học máy thường gồm các bước:

1. **Thu thập dữ liệu (Data Collection):** Lấy dữ liệu từ nhiều nguồn (cơ sở dữ liệu, hình ảnh, âm thanh, web…) và chuẩn bị định dạng phù hợp.

2. **Tiền xử lý dữ liệu (Preprocessing):** Làm sạch dữ liệu, loại bỏ dữ liệu trùng/thiếu, chuẩn hóa, biến đổi để phù hợp mô hình.
3. **Đào tạo mô hình (Training):** Chọn thuật toán phù hợp, dùng dữ liệu đã chuẩn bị để huấn luyện mô hình học các mẫu, mối quan hệ.
4. **Đánh giá mô hình (Evaluating):** Dùng tập dữ liệu kiểm tra (chưa dùng trong huấn luyện) để đo hiệu suất mô hình, xem mô hình có tổng quát tốt không.
5. **Điều chỉnh & tối ưu hóa tham số (Tuning):** Tinh chỉnh siêu tham số (hyperparameters) như learning rate, số lớp… để cải thiện hiệu suất.

2.3.4. Tổng quan học sâu

Học sâu (Deep Learning – DL) là một nhánh chuyên biệt của học máy (Machine Learning – ML), tập trung vào việc xây dựng và huấn luyện các mạng nơ-ron nhiều lớp (Deep Neural Networks – DNN).

Khác với các thuật toán truyền thống, DL cho phép tự động học các đặc trưng từ dữ liệu thô, đặc biệt hữu ích với dữ liệu phức tạp như hình ảnh, video, âm thanh và văn bản.

Học sâu đã chứng minh hiệu quả vượt trội trong nhiều ứng dụng:

- Computer Vision: nhận diện vật thể, phân đoạn ảnh, phát hiện hành động.
- Natural Language Processing: dịch máy, tóm tắt văn bản, sinh câu.
- Multimodal Tasks: Video Captioning, Image Captioning, Audio-to-Text.

2.3.5. Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN)

Mạng nơ-ron nhân tạo (Artificial Neural Networks – ANN) là một mô hình tính toán được lấy cảm hứng từ cấu trúc và chức năng của bộ não con người.

ANN được xây dựng dựa trên các nguyên tắc của sinh học. Mỗi nơ ron trong mạng hoạt động giống như một tế bào thần kinh trong não, tiếp nhận tín hiệu đầu vào, xử lý chúng và sau đó truyền đi tín hiệu đầu ra cho các nơ ron khác. Quá trình này giúp mạng học hỏi từ dữ liệu và cải thiện hiệu suất theo thời gian.

Cấu trúc của một mạng nơ ron nhân tạo thường bao gồm ba loại lớp: lớp đầu vào, lớp ẩn và lớp đầu ra. Mỗi lớp có vai trò riêng biệt, cộng tác với nhau để thực hiện những nhiệm vụ phức tạp.

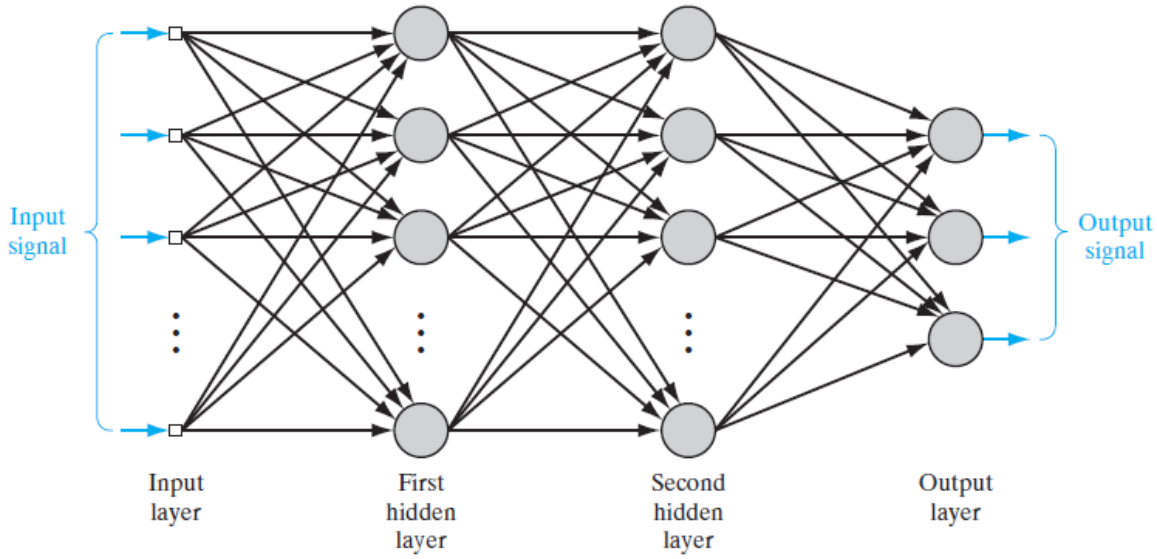


Figure 2-9: Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo gồm các đơn vị nhỏ gọi là nơ-ron, mỗi nơ-ron như một hàm tính toán. Một mạng nơ-ron nhân tạo bao gồm:

1. Lớp đầu vào (Input Layer)

– Nhận thông tin từ bên ngoài vào mạng.

– Mỗi nút trong lớp này đại diện cho một đặc trưng của dữ liệu đầu vào.

– Số lượng nơ-ron phụ thuộc vào số đặc trưng (features) của dữ liệu.

2. Lớp ẩn (Hidden Layer)

– Thực hiện các tính toán phức tạp để tìm đặc trưng ẩn trong dữ liệu nhằm chuyển đổi dữ liệu từ lớp đầu vào thành thông tin có thể hiểu được cho lớp đầu ra.

– Số lượng lớp ẩn có thể thay đổi tùy thuộc vào độ phức tạp của bài toán; càng nhiều lớp ẩn thì mạng càng có khả năng học được các mẫu phức tạp hơn.

– Có thể có nhiều lớp ẩn khác nhau.

3. Lớp đầu ra (Output Layer)

– Nơi đưa ra kết quả cuối cùng sau khi xử lý thông tin.

– Các nơ ron trong lớp này sẽ tổng hợp các tín hiệu từ lớp ẩn và cung cấp dự đoán hoặc phân loại cho các dữ liệu đầu vào.

– Số lượng nơ-ron phụ thuộc vào bài toán:

+ Phân loại nhị phân: 1 nơ-ron.

+ Phân loại đa lớp: Số nơ-ron tương ứng với số lớp.

a) Nguyên lý hoạt động:

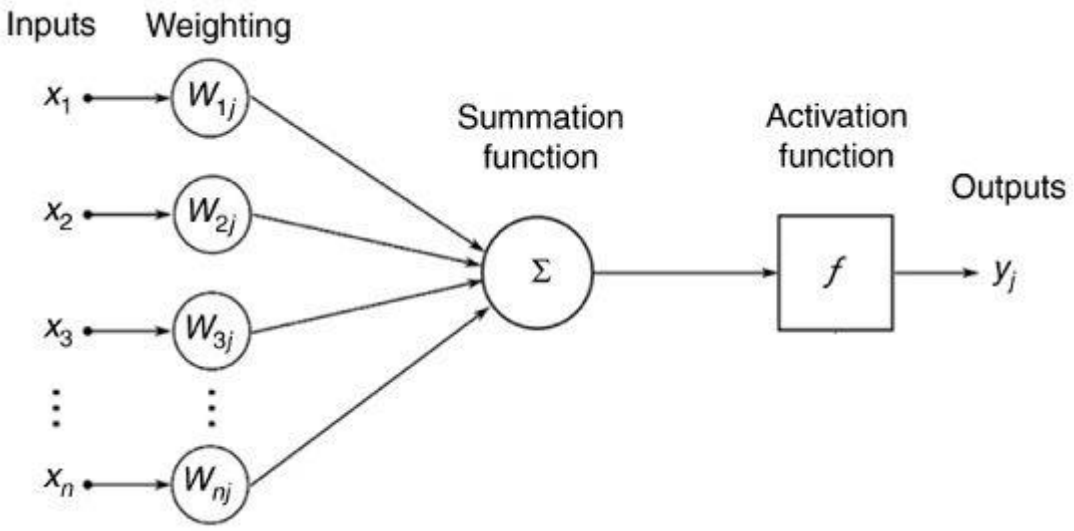


Figure 2-10: Nguyên lý hoạt động của mạng ANN

Mỗi nơ-ron thực hiện phép toán tuyến tính sau đó áp dụng hàm kích hoạt phi tuyến:

$$z = \sum_{i=1}^{n} w_i x_i + b, a = \phi(z)$$

Trong đó:

– x_i Là đầu vào, w_i là trọng số, b là bias.

– $\phi(\cdot)$ là hàm kích hoạt (activation function).

b) Các hàm kích hoạt phổ biến

– Hàm Sigmoid:

+ Chuyển đổi giá trị đầu vào thành giá trị từ 0 đến 1.

+ Phù hợp cho bài toán phân loại.

+ Công thức:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

– Hàm ReLU (Rectified Linear Unit):

+ Giúp mạng học nhanh hơn, đặc biệt trong mô hình lớn.

+ Công thức:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

– Hàm Tanh:

+ Tương tự Sigmoid, nhưng đầu ra từ -1 đến 1.

+ Hữu ích cho bài toán liên quan giá trị âm.

+ Công thức:

$$\mathrm{ReLU}(z) = \max(0, z)$$

– Hàm Softmax: thường dùng ở lớp cuối cho phân loại đa lớp:

$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, i = 1, \dots, C$$

c) Hàm mất mát (Loss Function)

Trong học sâu, hàm mất mát (Loss Function) — còn gọi là hàm chi phí (Cost Function) — là một thành phần cốt lõi giúp đánh giá độ sai lệch giữa giá trị dự đoán của mô hình và giá trị thực tế (ground truth).

Hàm mất mát là thước đo định lượng chất lượng dự đoán của mô hình:

– Nếu giá trị mất mát nhỏ → mô hình dự đoán gần đúng với thực tế.

– Nếu giá trị mất mát lớn → mô hình dự đoán sai lệch nhiều.

Khi huấn luyện mạng nơ-ron, mục tiêu của quá trình học là tối thiểu hóa giá trị của hàm mất mát thông qua thuật toán lan truyền ngược (backpropagation) và tối ưu hóa (optimization).

Hàm mất mát đo độ sai lệch giữa dự đoán $\hat{y}$ và nhãn thật y. Một số hàm phổ biến:

Mean Squared Error (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

– Phạt mạnh các sai số lớn do bình phương sai lệch.

- Phù hợp khi dữ liệu không có nhiễu lớn.
- Ứng dụng: hồi quy tuyến tính, mạng dự đoán giá trị liên tục.

Mean Absolute Error (MAE)

$$L = \frac{1}{n}\sum_{i=1}^{n} \mid y_i - \widehat{y_i} \mid$$

- Ít nhạy cảm với sai số lớn hơn MSE.
- Gradient không liên tục tại 0 nhưng dễ tính toán.
- Ứng dụng: mô hình hồi quy chịu nhiễu, hoặc bài toán yêu cầu tính bền vững cao.

Huber Loss:

$$L = \begin{cases} \frac{1}{2}(y_i - \widehat{y_i})^2, & \text{nếu } \mid y_i - \widehat{y_i} \mid \leq \delta \\ \delta(\mid y_i - \widehat{y_i} \mid - \frac{1}{2}\delta), & \text{nếu } \mid y_i - \widehat{y_i} \mid > \delta \end{cases}$$

- Kết hợp ưu điểm của MSE và MAE.
- Không quá nhạy với nhiễu, đồng thời vẫn mượt mà khi sai số nhỏ.
- Ứng dụng: bài toán hồi quy có outlier (nhiễu dữ liệu).

d) Lan truyền thuận (Forward Propagation)

Lan truyền thuận là quá trình mà dữ liệu đầu vào được truyền qua toàn bộ các lớp của mạng nơ-ron để tạo ra đầu ra dự đoán. Ở mỗi lớp, đầu vào được nhân với trọng số, cộng với hệ số bias, sau đó đi qua một hàm kích hoạt phi tuyến để sinh ra đầu ra.

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$
$$\boldsymbol{a}^{(l)} = \boldsymbol{f}(\boldsymbol{z}^{(l)})$$

Trong đó:

- $\boldsymbol{W}^{(l)}$: ma trận trọng số của lớp thứ $\boldsymbol{l}$

- $\boldsymbol{b}^{(l)}$: vector bias
- $\boldsymbol{a}^{(l-1)}$: đầu ra của lớp trước (với $\boldsymbol{a}^{(0)} = \boldsymbol{x}$là dữ liệu đầu vào)
- $\boldsymbol{f}(\cdot)$: hàm kích hoạt (ví dụ: ReLU, Sigmoid, Tanh, Softmax)

Diễn giải:

- Dữ liệu đầu vào được xử lý dần qua các tầng mạng, từ lớp đầu tiên đến lớp cuối cùng.
- Các lớp tích chập (convolution) trích xuất đặc trưng, lớp gộp (pooling) giảm kích thước dữ liệu, và các lớp kết nối đầy đủ (fully connected) thực hiện tổng hợp và suy luận.
- Kết quả cuối cùng $\boldsymbol{a}^{(L)}$là đầu ra dự đoán của mô hình (ví dụ: xác suất thuộc từng lớp).

Mục tiêu: Kết quả của lan truyền thuận sẽ được so sánh với giá trị thực tế để tính hàm mất mát, phục vụ bước lan truyền ngược tiếp theo.

e) Lan truyền ngược (Backward Propagation)

Lan truyền ngược là quá trình tính toán độ sai lệch (gradient) giữa đầu ra dự đoán và giá trị thực tế, sau đó truyền ngược lỗi qua các lớp để điều chỉnh trọng số và bias. Đây là cơ chế giúp mạng "học" được cách tối ưu hóa các tham số nhằm giảm sai số trong quá trình huấn luyện.

Đối với lớp cuối cùng $\boldsymbol{L}$:

$$\boldsymbol{\delta}^{(L)} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{a}^{(L)}} \odot \boldsymbol{f}'(\boldsymbol{z}^{(L)})$$

Đối với các lớp ẩn:

$$\boldsymbol{\delta}^{(l)} = ((\boldsymbol{W}^{(l+1)})^{\boldsymbol{T}} \boldsymbol{\delta}^{(l+1)}) \odot \boldsymbol{f}'(\boldsymbol{z}^{(l)})$$

Cập nhật trọng số và bias:

$$\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \boldsymbol{\eta} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}^{(l)}}$$

$$\boldsymbol{b}^{(l)} := \boldsymbol{b}^{(l)} - \boldsymbol{\eta} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{b}^{(l)}}$$

Trong đó:

- $\boldsymbol{\delta}^{(l)}$: sai số tại lớp $\boldsymbol{l}$
- $\boldsymbol{f}'(\boldsymbol{z}^{(l)})$: đạo hàm của hàm kích hoạt
- $\boldsymbol{\eta}$: tốc độ học (learning rate)
- $\boldsymbol{L}$: giá trị của hàm mất mát

Diễn giải:

- Sai số được truyền ngược từ lớp đầu ra về các lớp trước theo quy tắc đạo hàm chuỗi (chain rule).
- Mỗi lớp học được mức độ ảnh hưởng của trọng số đến sai số đầu ra, từ đó điều chỉnh các trọng số để giảm lỗi ở lần lặp tiếp theo.

Mục tiêu: Quá trình lan truyền ngược giúp tối ưu các tham số $\boldsymbol{W}$ và $\boldsymbol{b}$ sao cho hàm mất mát $\boldsymbol{L}$ đạt giá trị nhỏ nhất.

2.4. Mô hình EfficientNetB0

2.4.1. Giới thiệu EfficientNet

EfficientNet là một họ mô hình mạng nơ-ron tích chập (CNN) được đề xuất bởi Google vào năm 2019, với mục tiêu đạt được hiệu suất nhận dạng ảnh cao nhưng vẫn đảm bảo tối ưu về tài nguyên tính toán. Trước EfficientNet, các mô hình CNN thường được mở rộng theo từng chiều riêng lẻ như tăng chiều sâu (ResNet), tăng chiều rộng, hoặc tăng độ phân giải đầu vào. Tuy nhiên, việc mở rộng không cân bằng thường dẫn đến mô hình cồng kềnh hoặc không đạt hiệu quả mong muốn. EfficientNet khắc phục vấn đề này bằng cách giới thiệu chiến lược "Compound Scaling" – giúp mở rộng mô hình một cách cân bằng theo ba chiều depth, width và resolution. Đây là đóng góp quan trọng giúp EfficientNet đạt kết quả vượt trội trên ImageNet nhưng số tham số lại ít hơn đáng kể so với các mô hình CNN truyền thống.

2.4.2. Ý tưởng chính của EfficientNet

a) Compound Scaling

Compound Scaling là chiến lược mở rộng mô hình bằng cách kết hợp đồng thời ba yếu tố:

– Depth: tăng số lượng lớp → mô hình học đặc trưng sâu hơn.

– Width: tăng số lượng kênh mỗi lớp → mô hình học nhiều đặc trưng hơn.

– Resolution: tăng độ phân giải ảnh đầu vào → giữ được nhiều thông tin hơn.

Thay vì tăng từng yếu tố riêng rẽ, EfficientNet sử dụng một hệ số chung (phi) để mở rộng đồng thời và cân bằng cả ba chiều theo một công thức tối ưu hóa. Nhờ đó, mô hình đạt hiệu suất tốt hơn mà không tăng chi phí tính toán quá nhiều.

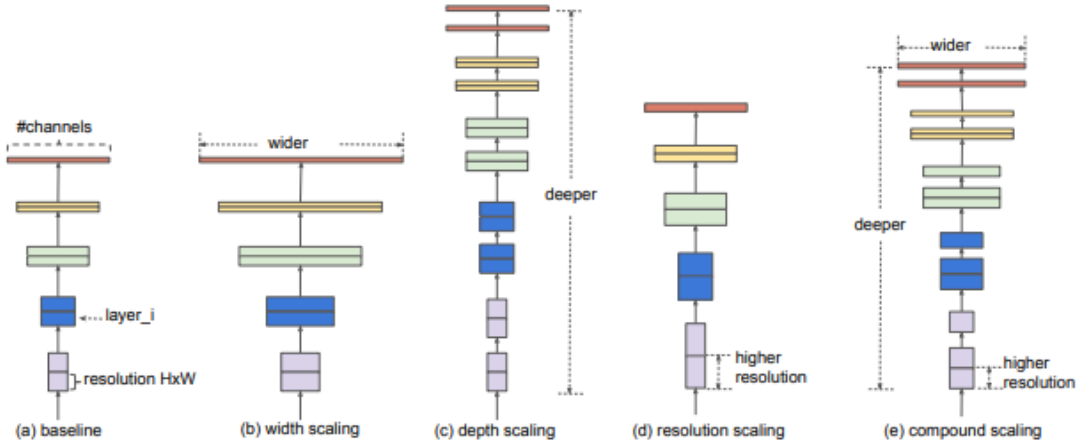


Figure 2-11: Minh họa compound scaling

b) Mobile Inverted Bottleneck Convolution (MBConv)

EfficientNet sử dụng các khối MBConv, vốn được giới thiệu trong MobileNetV2. MBConv có cấu trúc "inverted bottleneck" với các đặc điểm:

– Mở rộng số kênh trước, sau đó nén lại để tiết kiệm tính toán.

– Sử dụng depthwise convolution giúp giảm số lượng tham số rất lớn.

– Tối ưu cho thiết bị di động và hệ thống tài nguyên thấp.

Việc sử dụng MBConv giúp EfficientNetB0 nhỏ gọn nhưng vẫn giữ được khả năng học đặc trưng mạnh.

c) Squeeze-and-Excitation (SE)

SE block được tích hợp vào MBConv để tăng khả năng chú ý kênh (channel attention).

Cơ chế hoạt động gồm 2 bước:

– Squeeze: tóm tắt thông tin của từng kênh

– Excitation: học trọng số quan trọng cho mỗi kênh

Nhờ SE block, mô hình biết kênh nào quan trọng để tập trung vào, giúp tăng accuracy.

2.4.3. Kiến trúc EfficientNetB0

EfficientNetB0 là phiên bản cơ sở (baseline) của cả họ EfficientNet. Kiến trúc gồm:

- Lớp Convolution đầu vào
- Các nhóm MBConv1 và MBConv6 với số lượng block khác nhau
- Các kích thước filter thay đổi theo từng giai đoạn
- Cuối cùng là Global Average Pooling → Fully Connected → Softmax

EfficientNetB0 có khoảng **5.3 triệu tham số**, nhẹ hơn rất nhiều so với các mô hình như ResNet50 (~25M), nhưng lại đạt accuracy cao hơn trên ImageNet. Đây là lý do EfficientNetB0 rất phù hợp cho các hệ thống có hạn chế phần cứng như Raspberry Pi, ESP32-CAM hoặc máy tính cấu hình thấp.

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

Figure 2-12: Kiến trúc EfficientNetB0

2.5. Mô hình Resnet50

2.5.1. Giới thiệu mô hình

Trí tuệ nhân tạo (AI) đang thay đổi mạnh mẽ cách chúng ta xử lý và phân tích dữ liệu hình ảnh, từ nhận diện đối tượng đến phân loại ảnh y tế. Tuy nhiên, việc triển khai các mô hình AI sâu với độ chính xác cao thường gặp thách thức về hiệu năng và chi phí tính toán, đặc biệt với các tập dữ liệu lớn.

Trong bối cảnh đó, ResNet-50 nổi bật như một kiến trúc mạng sâu hiệu quả, cung cấp khả năng học các đặc trưng phức tạp trong khi giảm thiểu vấn đề biến mất gradient nhờ cơ chế residual (dạng khối dư). ResNet-50 là phiên bản 50 lớp của dòng ResNet,

được thiết kế để đạt độ chính xác cao trên các tác vụ phân loại ảnh mà vẫn giữ được khả năng huấn luyện ổn định.

Kiến trúc này sử dụng các khối residual, cho phép tín hiệu đầu vào "bỏ qua" một số tầng mạng khi cần thiết, giúp mạng học các đặc trưng sâu mà không làm suy giảm thông tin. Ngoài ra, ResNet-50 có thể kết hợp với học chuyển giao, giúp tận dụng các trọng số đã huấn luyện trước trên các tập dữ liệu lớn như ImageNet, từ đó tăng tốc quá trình huấn luyện và cải thiện hiệu quả trên các tập dữ liệu nhỏ hoặc chuyên biệt.

Những đặc điểm này khiến ResNet-50 trở thành lựa chọn lý tưởng cho các hệ thống yêu cầu độ chính xác cao trong phân loại ảnh, nhận diện đối tượng, hoặc các ứng dụng AI phức tạp trên nền tảng máy chủ hoặc các hệ thống có khả năng tính toán mạnh.

2.5.2. Vanishing Gradient

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

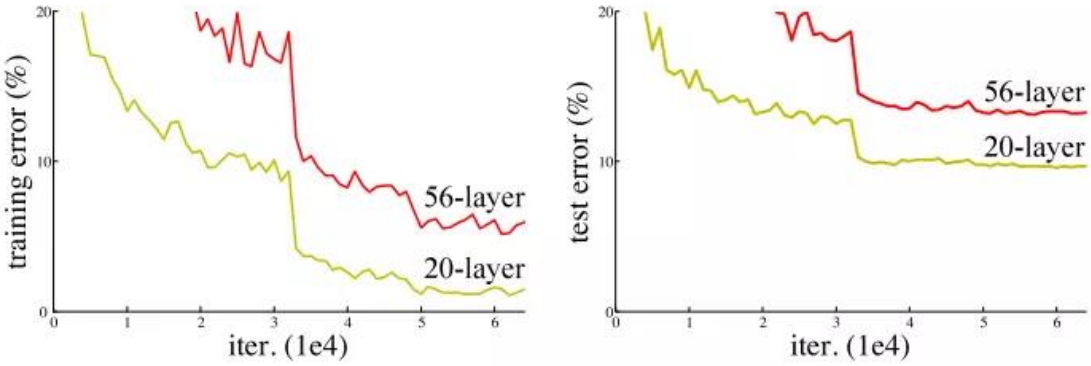


Figure 2-13: Minh họa huấn luyện mạng CNN

Khi huấn luyện mạng nơ-ron sâu, Backpropagation tính gradient của hàm mất mát để cập nhật tham số qua Gradient Descent. Tuy nhiên, ở các lớp sâu, gradient thường giảm dần, khiến các tham số ít thay đổi và mạng khó hội tụ — hiện tượng này gọi là vanishing gradient.

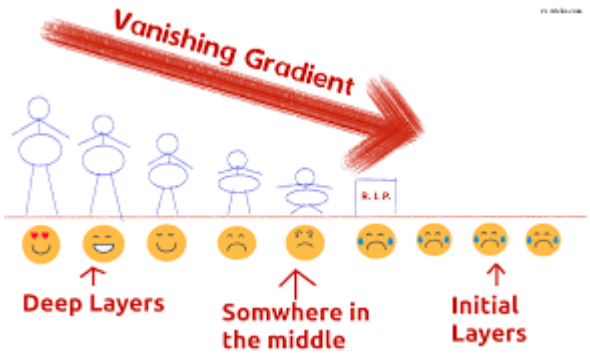


Figure 2-14: Vanishing Gradient

ResNet giải quyết vấn đề này nhờ residual connections, cho phép gradient truyền thẳng qua các lớp sâu, cải thiện khả năng học hiệu quả của mạng.

2.5.3. Residual Block

ResNet giải quyết vấn đề vanishing gradient bằng cách sử dụng kết nối tắt (shortcut connection), cho phép truyền thông tin trực tiếp qua một hoặc nhiều lớp. Mỗi khối như vậy được gọi là Residual Block.

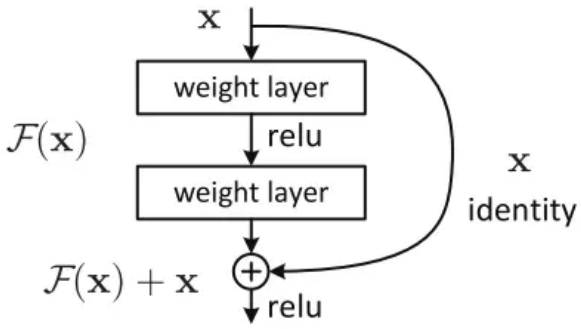


Figure 2-15: Residual Block

Về cơ bản, ResNet vẫn bao gồm các lớp convolution, pooling, activation và fully-connected như các mạng CNN thông thường. Điểm khác biệt là đầu vào của khối được cộng trực tiếp vào đầu ra của các layer trong khối, như minh họa trong Residual Block:

$$H(x) = F(x) + x$$

Trong đó:

- x là đầu vào của khối.
- $F(x)$ là output của các layer bên trong khối (ví dụ: $x \rightarrow weight_1 \rightarrow ReLU \rightarrow weight_2$)
- $H(x)$ là đầu ra cuối cùng sau khi cộng với x và đi qua ReLU

Cơ chế này giúp gradient có đường đi trực tiếp, chống lại hiện tượng gradient biến mất, đồng thời cải thiện khả năng học của mạng sâu.

2.5.4. Khả năng mở rộng mạng sâu với ResNet

Trong các mạng CNN truyền thống, việc tăng số lượng lớp thường dẫn đến giảm độ chính xác do vấn đề vanishing gradient. Tuy nhiên, ResNet đã chứng minh rằng mạng sâu vẫn có thể học hiệu quả nhờ các kết nối tắt (shortcut connections).

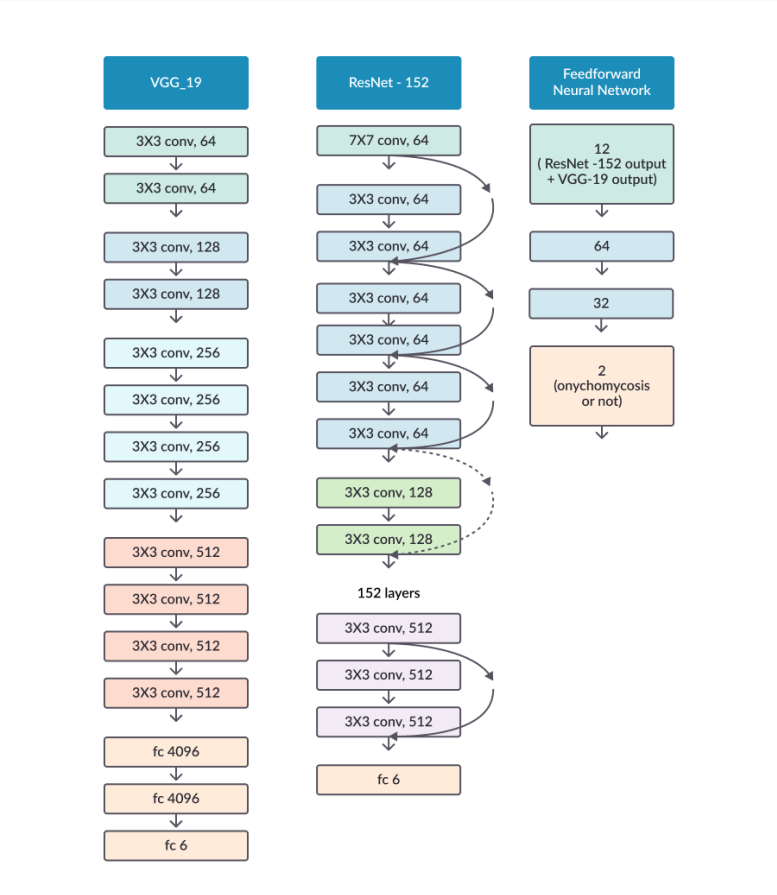


Figure 2-16: Các mô hình CNN

- **Hình 1 :** VGG-19 sử dụng kernel 3×3 trên toàn bộ mạng và đã giành chiến thắng tại ILSVRC 2014.
- **Hình 2 :** ResNet thêm các kết nối tắt, nối trực tiếp đầu vào của lớp n với lớp $n + x$ (biểu diễn bằng mũi tên cong), giúp cải thiện hiệu suất huấn luyện ngay cả với mô hình hơn 20 lớp.
- **Hình 3 :** Tổng cộng 12 đầu ra từ ResNet-152 và VGG-19 được đưa vào mạng có 2 lớp hidden. Các lớp ẩn này tính toán đầu ra cuối cùng, cho thấy việc xếp

chồng lớp không làm giảm hiệu suất. Kiến trúc này giúp các lớp phía trên nhận thông tin trực tiếp từ các lớp dưới, từ đó điều chỉnh trọng số hiệu quả hơn.

Nhờ cơ chế này, ResNet cho phép xây dựng các mạng rất sâu mà vẫn duy trì hiệu năng cao, vượt trội so với các kiến trúc CNN truyền thống.

2.5.5. Kiến trúc mạng ResNet50

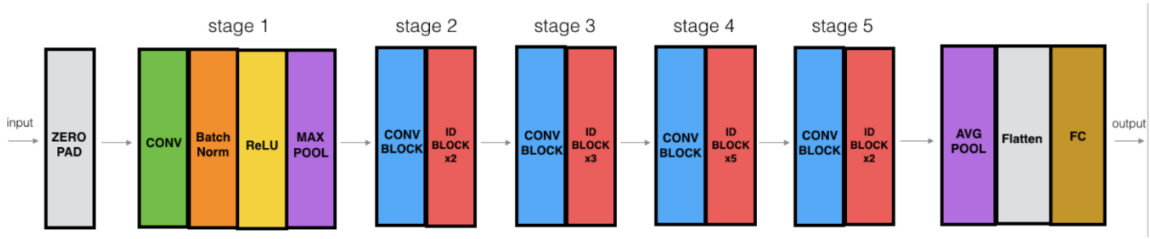


Figure 2-17: Kiến trúc mạng ResNet50

ResNet-50 bao gồm các Residual Block được xếp chồng, kết hợp với các lớp convolution, pooling, batch normalization và fully connected, cho phép mạng sâu nhưng vẫn duy trì khả năng huấn luyện hiệu quả. Cấu trúc chi tiết gồm các stage như sau:

Input và Zero-padding: Input có kích thước (3×3) được zero-padding.

Stage 1:

- Convolutional layer (Conv1) với 64 filter, kích thước 7×7, stride 2×2.
- Batch Normalization và MaxPooling 3×3.

Stage 2:

- Convolutional block với 3 filter kích thước 64×64×256, f=3, s=1.
- 2 Identity blocks với filter size 64×64×256, f=3.

Stage 3:

- Convolutional block với 3 filter kích thước 128×128×512, f=3, s=2.
- 3 Identity blocks với filter size 128×128×512, f=3.

Stage 4:

- Convolutional block với 3 filter kích thước 256×256×1024, f=3, s=2.
- 5 Identity blocks với filter size 256×256×1024, f=3.

Stage 5:

- Convolutional block với 3 filter kích thước 512×512×2048, f=3, s=2.

- 2 Identity blocks với filter size 512×512×2048, f=3.

Pooling và Fully Connected:

- 2D Average Pooling với kích thước 2×2.
- Flatten layer.
- Fully Connected (Dense) với softmax activation để dự đoán lớp đầu ra.

Chú thích:

- Identity Block (ID BLOCK): Là khối residual, đầu vào được cộng trực tiếp với output của các layer bên trong khối. Khi ghi "ID BLOCK x3" nghĩa là có 3 khối Identity block chồng lên nhau.

Cấu trúc này cho phép ResNet-50 học được các đặc trưng phức tạp, giảm thiểu vấn đề vanishing gradient và đạt hiệu suất cao trên các tập dữ liệu lớn như ImageNet.

2.6. Mô hình MobileNet

2.6.1. Giới thiệu mô hình

MobileNet xuất hiện như một giải pháp đột phá, mang đến sự cân bằng giữa độ chính xác và hiệu quả tính toán, mở ra cơ hội mới cho các ứng dụng AI trên các thiết bị di động và các hệ thống nhúng. MobileNetV3Large là mô hình được quan tâm cho học chuyển giao và đây là phiên bản thứ ba của lớp mô hình MobileNet.

Để giữ cho thiết kế gọn nhẹ, MobileNetV3Large sử dụng nhiều kỹ thuật, bao gồm phần dư đảo ngược, hàm kích hoạt mở rộng, tìm kiếm kiến trúc mạng, các khối nén và kích thích, và một đầu phân loại nhẹ.

2.6.2. Standard Convolutions

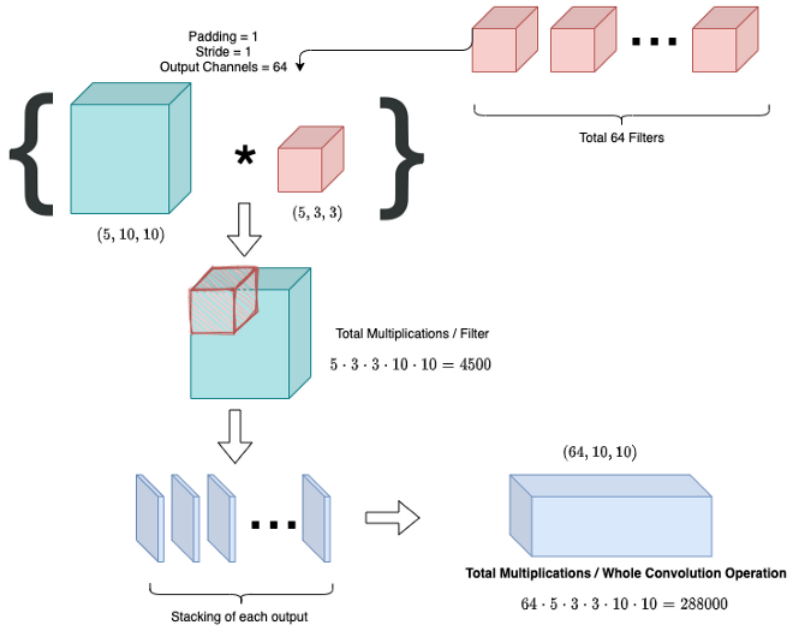


Figure 2-18: Standard Convolutions

Chú thích: (5, 10, 10) = (số channel, width, height). Nhìn vào hình, ta có thể thấy:

- Đầu vào của chúng ta là 1 feature map 5x10x10. Bộ lọc của chúng ta có kích thước 5x3x3. Như vậy, sau khi thực hiện phép nhân tích chập trên toàn bộ feature map (padding = 1, stride=1) ta thu được kết quả 1x10x10.
- Với mỗi điểm trên feature map, vì filter là 3x3, nên cần thực hiện 3x3 = 9 phép tính nhân.
- Số tính toán đã thực hiện là 5 x (3 x 3 x 10 x 10) = 4500.
- Sau khi thực hiện phép nhân, ta cộng nó lại. Nhưng đừng lo về phép cộng. Với máy tính 64-bit, thời gian thực hiện phép nhân có thể gấp 64 lần thời gian thực hiện phép cộng, well, có nghĩa là không đáng kể ở đây.
- Trong một lớp convolutional, ta sẽ sử dụng nhiều bộ lọc. Ở đây ta có 64 bộ lọc, vậy tổng số tính toán cần thực hiện là : 64 x 4500 = 288000 phép tính nhân.

Như vậy, với:

- M, N Là số input, output channels.
- D_f Là chiều của input feature map ($M * D_f * D_f$).
- D_k Là chiều của kernel.

Thì tổng số phép nhân cần thực hiện là $M * N * D_f^2 * D_k^2$.

2.6.3. Kiến trúc MobileNet

d) Depthwise Convolutions

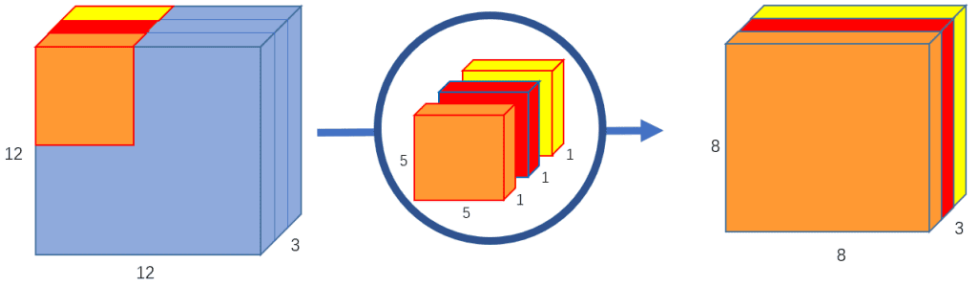


Figure 2-19: Depthwise Convolutions

Thay vì nhân một mớ tất cả ($M * N * D_f^2 * D_k^2$) thì ta tách ra một chút, thực hiện:

- Nhân tích chập 5x10x10 với bộ filter giờ chỉ còn là 1x3x3, tương tự với 5 filters như thế, stack nó lại, kết quả thu được output là 5x10x10.
- Số phép nhân đã thực hiện vẫn là $M * D_f^2 * D_k^2$ = 5 x 10 x 10 x 3 x 3 = 4500 phép nhân.

Tuy nhiên, có thể thấy khác với standard CNN, ở deepwise convolutions, số lượng channel của chúng ta vẫn giữ nguyên, nghĩa là vẫn 5x10x10.

e) Pointwise Convolution:

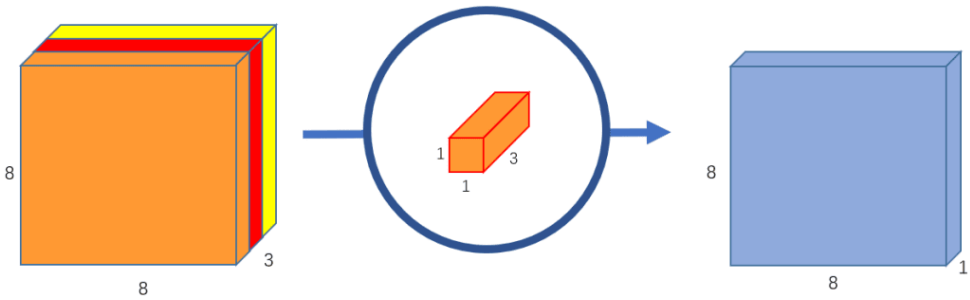


Figure 2-20: Pointwise Convolution

Sử dụng kết quả từ bước deepwise convolution. Ở bước pointwise này, ta chỉ sử dụng bộ có kích thước là 1x1. Đồng thời số lượng bộ lọc bằng số channel mà ta muốn thu được. Ta muốn tăng lên 64 channel, vậy hãy sử dụng 64 bộ filters.

Số phép nhân cần tính chỉ là $M * N * D_f^2$ = 5 x 64 x 10 x 10 = 32000 phép nhân, tổng số phép nhân cần tính là: 4500 + 32000 = 36500 phép nhân, số lượng tính toán đã giảm 8 lần so với standard CNN.

Số lượng tính toán đã giảm được là:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

Hình dưới cho thấy sự khác biệt giữa standard CNN và kiến trúc MobileNet:

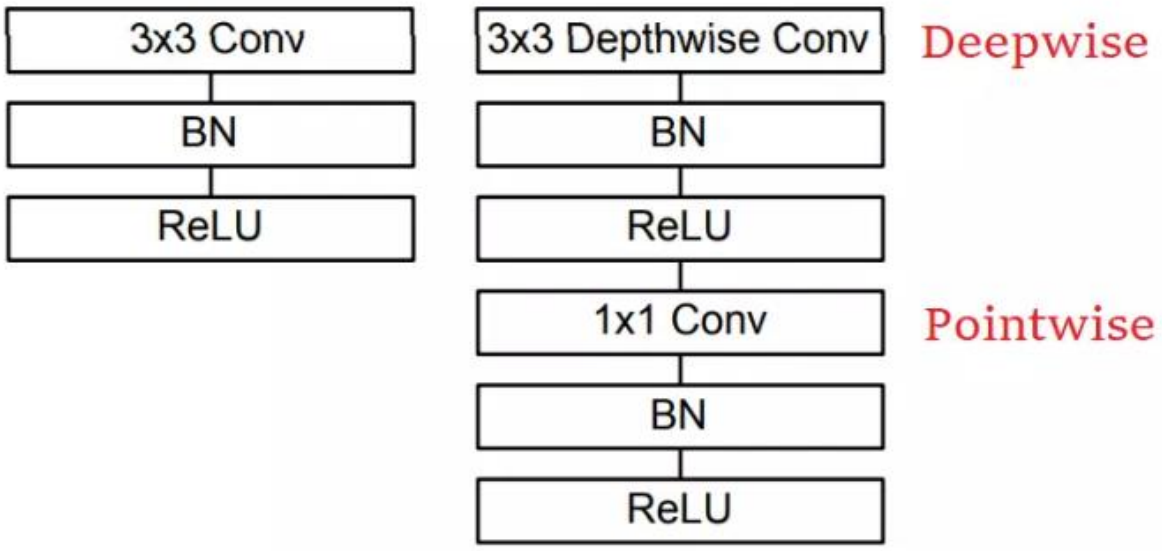


Figure 2-21: So sánh stardard CNN và MobileNet

f) Width Multiplier: Thinner Models

Width Multiplier là một tham số được MobileNet giới thiệu để điều chỉnh độ rộng của mạng, tức là số lượng kênh đầu ra tại mỗi lớp. Thay vì sử dụng số lượng kênh chuẩn (ví dụ: 32, 64, 128, v.v.), MobileNet cho phép điều chỉnh số lượng kênh này bằng cách nhân với một hệ số alpha ($0 < \alpha \leq 1$). Điều này giúp giảm số lượng tham số và phép tính cần thiết mà không ảnh hưởng quá nhiều đến độ chính xác của mô hình.

Width Multiplier α được thêm vào 1 layer, giờ đây số input channels và output channels là αM và αN, thay vì M và N. Chi phí tính toán của một phép tích chập là:

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

có nghĩa là α càng nhỏ thì chi phí tính toán càng giảm.

g) Resolution Multiplier: Reduced Representation

Tương tự như Width Multiplier α, Resolution Multiplier ρ cũng được thêm vào các layer để giảm resolution của ảnh đầu vào và các biểu diễn bên trong giữa các layers.

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

với $\rho \in (0,1]$, có nghĩa là ρ càng nhỏ thì chi phí tính toán càng giảm.

2.6.4. Mô hình cải tiến MobileNetV2 và MobileNetV3

a) Tổng quan

MobileNetV3 tiếp tục sử dụng Depthwise Separable Convolutions của MobileNetV1, ngoài ra còn sử dụng những phương pháp đề xuất của MobileNetV2:

- Linear bottlenecks
- Inverted Residual Block (shortcut connections giữa các bottlenecks)

Và điểm cải tiến chính là việc bổ sung **Squeeze-and-Excite**.

Nhìn chút về sự khác nhau của V1 và V2:

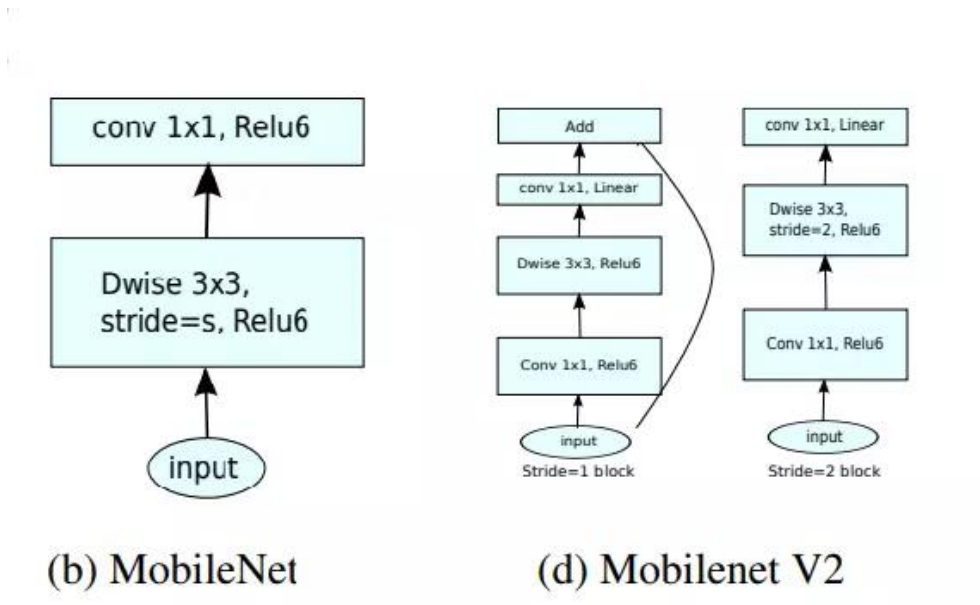


Figure 2-22: So sánh MobileNetV1 và V2

Hãy phân tích sự khác biệt trong bức ảnh trên:

- MobileNet gồm sử dụng 1 loại blocks gồm 2 phần, Deepwise và Pointwise.

– MobileNet v2 sử dụng 2 loại blocks, bao gồm: residual block với stride = 1 và block với stride = 2 phục vụ downsizing.

Có 3 phần đối với mỗi block:

– Layer đầu là 1×1 convolution với ReLU6.

– Layer thứ hai, như cũ, là depthwise convolution.

– Layer thứ 3 tiếp tục là 1×1 convolution nhưng không có activation function. Linear được sử dụng thay vì ReLu như bình thường.

Kết nối tắt ở MobileNetV2 được điều chỉnh sao cho số input và output channels của mỗi block residual được thắt hẹp lại. Chính vì thế nó được gọi là các bottleneck layers.

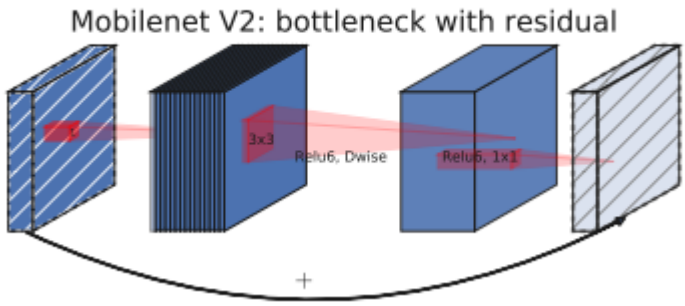


Figure 2-23: Kiến trúc MobileNetV2

Residual block của MobileNet v2 ngược lại so với các kiến trúc residual truyền thống, vì kiến trúc residual truyền thống có số lượng kênh ở input và output của một block lớn hơn so với các layer trung gian, gọi là **Inverted residual block**.

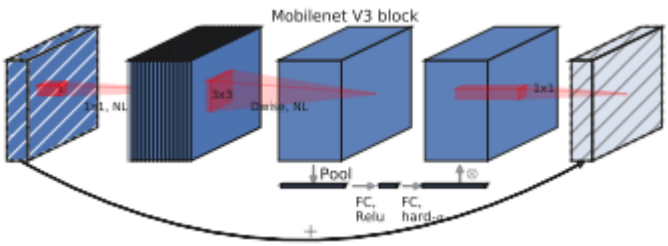


Figure 2-24: Kiến trúc MobileNetV3

Đối với MobileNetV3, ở đây sử dụng sự kết hợp của các lớp này làm các khối xây dựng để xây dựng các mô hình hiệu quả nhất. Các lớp cũng được nâng cấp với các phi tuyến swish đã được chỉnh sửa. Cả squeeze and excitation và phi tuyến swish đều sử

dụng sigmoid, điều này có thể không hiệu quả trong tính toán và khó duy trì độ chính xác trong số học số nguyên cố định, vì vậy họ thay thế bằng hard sigmoid.

b) Platform-Aware NAS and NetAdapt

Tóm tắt quy trình như sau:

- Bắt đầu với một kiến trúc mạng hạt giống được tìm thấy bằng platform-aware NAS.
- Đối với mỗi bước:
  + Tạo một tập hợp các đề xuất mới, mỗi đề xuất là một sửa đổi của kiến trúc nhằm giảm độ trễ.
  + Sử dụng mô hình đã được huấn luyện từ bước trước và áp dụng vào kiến trúc mới, cắt ngắn và khởi tạo ngẫu nhiên các trọng số thiếu khi cần. Tinh chỉnh từng đề xuất để ước lượng độ chính xác.
  + Chọn đề xuất tốt nhất dựa trên một số tiêu chí.
- Lặp lại bước trước cho đến khi đạt được độ trễ mục tiêu.

c) Redesigning Expensive Layers

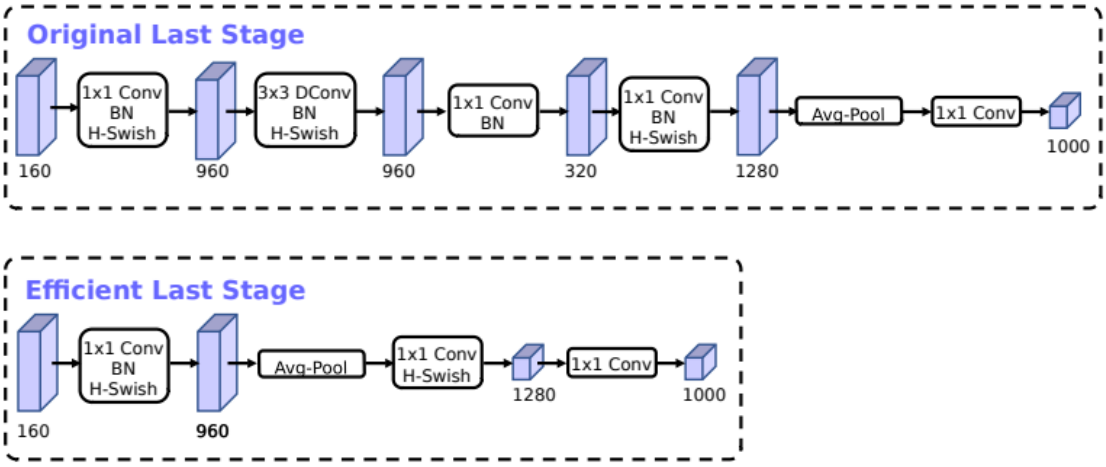


Figure 2-25: Kiến trúc tinh giản Layer

Efficient Last Stage đã tinh giản toàn bộ phần cuối, dựa trên NAS + NetAdapt:

- Chuyển lớp này qua cuối lớp average pooling.
- Tập hợp các đặc trưng cuối cùng này giờ đây được tính toán ở độ phân giải không gian 1x1 thay vì 7x7.

– Việc giảm số lượng bộ lọc và sử dụng các phi tuyến khác nhau để cố gắng giảm sự dư thừa, sử dụng phi tuyến hard swish cho lớp này.

2.7. Tăng cường dữ liệu

2.7.1. Điều chỉnh ảnh

Table 2-1: Điều chỉnh ảnh

Tham số	Công dụng
rescale	Chuẩn hóa giá trị pixel, thường dùng: rescale=1./255
rotation_range	Xoay ảnh ngẫu nhiên theo độ (VD: 20 ⇒ xoay ±20°)
width_shift_range	Tịnh tiến ảnh theo chiều ngang (VD: 0.1 ⇒ 10% chiều rộng)
height_shift_range	Tịnh tiến ảnh theo chiều dọc
zoom_range	Phóng to/thu nhỏ ảnh ngẫu nhiên
shear_range	Biến dạng cắt xiên ảnh (shear transformation)
brightness_range	Thay đổi mức sáng ảnh
horizontal_flip	Lật ngang ảnh ngẫu nhiên
vertical_flip	Lật dọc ảnh ngẫu nhiên
validation_split	Tự chia train/validation trong 1 dataset (VD: 0.2 ⇒ 20% validation)

2.7.2. Phương pháp MixUp

Trộn hai hình ảnh và nhãn của chúng với xác suất nhất định. Các mixup siêu tham số xác định xác suất áp dụng phép biến đổi, với mixup=1.0 đảm bảo rằng tất cả hình ảnh được trộn và mixup=0.0 vô hiệu hóa phép biến đổi. Ví dụ: với mixup=0.5, mỗi ảnh có 50% cơ hội được trộn với một ảnh khác.

Cải thiện độ mạnh mẽ của mô hình và giảm tình trạng overfitting. Ví dụ: trong các hệ thống nhận dạng sản phẩm bán lẻ, mixup giúp mô hình học các đặc trưng mạnh mẽ hơn bằng cách trộn lẫn hình ảnh của các sản phẩm khác nhau, dạy nó cách xác định các mặt hàng ngay cả khi chúng chỉ hiển thị một phần hoặc bị che khuất bởi các sản phẩm khác trên các kệ hàng đông đúc.

Figure 2-26: Kĩ thuật MixUp

2.8. Các kĩ thuật tối ưu hóa huấn luyện

2.8.1. Class Weight

Trong nhiều bài toán phân loại, số lượng mẫu giữa các lớp chênh lệch lớn (ví dụ: 95% lớp A và 5% lớp B). Khi đó mô hình có xu hướng:

- Học thiên lệch về lớp chiếm đa số
- Ít quan tâm đến các mẫu hiếm
- Đạt accuracy cao nhưng thực chất dự đoán kém với lớp minor

Class Weight là một phương pháp **Cost-Sensitive Learning**, trong đó mỗi lớp được gán một trọng số khác nhau. Trọng số càng lớn thì mẫu của lớp đó càng "quan trọng" trong hàm loss.

Hàm loss có trọng số:

$$L = \sum_{i=1}^{N} w_{y_i} \cdot \ell(y_i, \hat{y}_i)$$

Trong đó:

- w_{y_i}: trọng số tương ứng lớp thật
- ℓ: cross-entropy hoặc một loss khác
- y_i: nhãn thực
- $\hat{y}_i$: dự đoán

Tính trọng số:

$$w_c = \frac{1}{\text{frequency}(c)}$$

Hoặc trong sklearn:

$$w_c = \frac{n_{samples}}{n_{classes} \cdot n_c}$$

2.8.2. Focal Loss

Trong bài toán detection, đặc biệt là phát hiện dense objects (nhiều bounding box), số lượng mẫu "dễ" (background) chiếm đa số. Điều này dẫn tới:

- Loss bị thống trị bởi các mẫu dễ
- Mô hình không tập trung vào các mẫu khó hoặc lớp hiếm

Cross-Entropy truyền thống (CE):

$$CE(p_t) = -\log(p_t)$$

Focal Loss (giảm ảnh hưởng của mẫu dễ):

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

Trong đó:

- p_t: xác suất mô hình dự đoán đúng
- γ: hệ số điều chỉnh (thường 2)

- Khi p_t lớn → mẫu dễ → $(1-p_t)^\gamma$ nhỏ → giảm trọng số
- Khi p_t nhỏ → mẫu khó → trọng số gần 1 → được học mạnh hơn

Focal Loss giảm độ quan trọng của các mẫu đã được phân loại chính xác, để mô hình không bị "phân tâm", giảm overfitting khi dữ liệu cực kỳ mất cân bằng.

2.9. Các chỉ số đánh giá mô hình

2.9.1. Accuracy (Đô chính xác)

Độ chính xác (Accuracy) là một chỉ số đánh giá mô hình phân loại, cho biết tỷ lệ mẫu mà mô hình dự đoán đúng so với tổng số mẫu.

Công thức:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Trong đó:

- TP (True Positive): dự đoán đúng lớp dương
- TN (True Negative): dự đoán đúng lớp âm
- FP (False Positive): dự đoán sai thành dương
- FN (False Negative): dự đoán sai thành âm

Ý nghĩa:

- Accuracy cao → mô hình dự đoán đúng nhiều
- Phù hợp khi dữ liệu cân bằng
- Không phù hợp khi dữ liệu mất cân bằng, vì chỉ cần đoán lớp chiếm đa số cũng đạt accuracy cao.

2.9.2. Loss (Hàm mất mát)

Loss phản ánh mức độ sai lệch giữa dự đoán của mô hình và giá trị thật. Loss được tính trong quá trình huấn luyện, giúp mô hình biết cần điều chỉnh trọng số theo hướng nào.

Cross-Entropy Loss (phân loại nhiều lớp):

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Binary Cross-Entropy (nhị phân):

$$L = -[y\log(p) + (1-y)\log(1-p)]$$

Ý nghĩa:

- Loss giảm đều → mô hình đang học tốt
- Loss giảm nhưng accuracy không tăng → mô hình học "quá tự tin" vào một số lớp
- Training loss giảm nhưng validation loss tăng → overfitting

2.9.3. Classification Report

Bao gồm 3 chỉ số quan trọng nhất: Precision, Recall, và F1-score. Đây là chỉ số chuẩn cho dữ liệu mất cân bằng.

Precision (Độ chính xác dương)

Precision cho biết khi mô hình dự đoán là lớp X, nó đúng bao nhiêu phần trăm.

$$Precision = \frac{TP}{TP + FP}$$

→ Precision cao nghĩa là mô hình ít dự đoán nhầm sang lớp khác.

Recall (Độ bao phủ)

Recall cho biết trong tất cả các mẫu thuộc lớp X, mô hình tìm được bao nhiêu.

$$Recall = \frac{TP}{TP + FN}$$

→ Recall cao nghĩa là mô hình bỏ sót ít mẫu thuộc lớp thật.

F1-score

F1-score là trung bình điều hòa giữa Precision và Recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

→ F1-score hữu ích cho dữ liệu không cân bằng vì nó cân bằng giữa "độ chính xác dương" và "độ bao phủ".

Ý nghĩa:

- Đánh giá mô hình cho từng lớp một cách chi tiết
- Phù hợp cho bài toán nhiều lớp và mất cân bằng
- Tốt hơn Accuracy rất nhiều, vì xem xét TP, FP, FN một cách độc lập

CHƯƠNG 3: DỮ LIỆU HUẤN LUYỆN VÀ TIỀN XỬ LÝ

3.1. Nguồn dữ liệu

Trong lĩnh vực **Flower Classification**, việc lựa chọn nguồn dữ liệu ảnh hưởng trực tiếp đến khả năng học đặc trưng hình thái, chất lượng mô hình và mức độ tổng quát hóa khi triển khai thực tế. Trong nghiên cứu này, dữ liệu được tổng hợp từ nhiều nguồn khác nhau nhằm đảm bảo tính đa dạng, tính đầy đủ và độ tin cậy của mẫu đầu vào. Các nguồn dữ liệu sử dụng bao gồm:

Bộ dữ liệu **Oxford 102**[1] là một trong những bộ dữ liệu tiêu chuẩn và được sử dụng rộng rãi nhất trong các nghiên cứu phân loại hoa bằng thị giác máy tính. Bộ dữ liệu gồm 102 loài hoa khác nhau, mỗi loài có nhiều ảnh chụp trong các điều kiện ánh sáng, góc chụp và nền cảnh đa dạng. Đây là nguồn dữ liệu có tính chuẩn hóa cao và được trích dẫn trong nhiều nghiên cứu tiên tiến (state-of-the-art), giúp kết quả của đề tài dễ dàng so sánh với các mô hình khác trong cùng lĩnh vực.

Để bổ sung thông tin mô tả và đặc điểm sinh học của các loài hoa phổ biến tại Việt Nam, nghiên cứu sử dụng dữ liệu từ **hoavietnam** [2]. Nguồn này cung cấp:

- Tên khoa học và tên tiếng việt thường dùng của hoa.
- Đặc điểm hình thái như màu sắc, cấu trúc cánh, lá, nhị, dạng cụm hoa.

Ngoài bộ dữ liệu chuẩn hóa Oxford 102, nghiên cứu còn mở rộng phạm vi bằng danh sách **60 loài hoa phổ biến** [3] được tổng hợp từ các nguồn thực vật học.

Những nguồn trên đã đảm bảo về chuẩn tên khoa học của các loài hoa, để đảm bảo độ chính xác khi phân loại và kiểm chứng đặc điểm hình ảnh, đề tài sử dụng nguồn ảnh từ cộng đồng **iNaturalist** [4]. Đây là nền tảng cung cấp ảnh thực địa chất lượng cao của sinh vật, trong đó có thực vật và hoa. Các ảnh trên iNaturalist được gán nhãn bởi cộng đồng chuyên gia và người dùng giàu kinh nghiệm, giúp tăng độ tin cậy khi:

- Đối chiếu đặc điểm của từng loài hoa.
- Xác định hình thái đặc trưng trong điều kiện tự nhiên.
- Hỗ trợ quá trình đánh giá mô hình trong các trường hợp mẫu ảnh nằm ngoài dữ liệu huấn luyện.

3.2. Quá trình thu thập dữ liệu

Quá trình thu thập dữ liệu cho bài toán phân loại hoa được thực hiện theo hai giai đoạn chính: (1) thu thập và chuẩn hóa thông tin tên khoa học của từng loài hoa, và (2) tự động thu thập ảnh minh họa từ nền tảng iNaturalist dựa trên danh sách đã tổng hợp.

3.2.1. Thu thập tên khoa học

Tên khoa học (scientific name) là yếu tố quan trọng để đảm bảo tính chính xác trong nhận dạng loài và là khóa tham chiếu thống nhất khi kết hợp dữ liệu từ nhiều nguồn khác nhau. Trong nghiên cứu này, danh sách tên khoa học được thu thập bằng phương pháp thủ công từ ba nhóm nguồn dữ liệu chính:

- **Bộ dữ liệu 102 loài hoa Oxford** – cung cấp tên khoa học chính xác và được chuẩn hóa.
- **Trang thông tin hoavietnam.com** – cung cấp tên tiếng Việt và tên khoa học tương ứng của các loài phổ biến tại Việt Nam.
- **Danh sách Top 60 loài hoa phổ biến** – bổ sung các loài mà Oxford 102 không có, kèm theo tên khoa học từ các tài liệu thực vật học và website đối chiếu.

Trong giai đoạn này, mỗi loài hoa được kiểm tra và xác nhận lại tên khoa học dựa trên ít nhất hai nguồn để giảm sai lệch. Sau khi hoàn tất, danh sách được tổng hợp thành một file CSV chuẩn hóa với cột **ten_khoa_hoc** đóng vai trò mã định danh duy nhất cho từng loài.

Kiểm tra thử **ten_khoa_hoc** loài thu thập được có trên iNaturalist:

```
Tổng số loài cần kiểm tra: 113

=== Kết quả kiểm tra iNaturalist ===
+---------------------------+---------------------------+---------------------------+-------+
| Tên khoa học              | Tồn tại trên iNaturalist  | Tên khớp                  | Lỗi   |
+===========================+===========================+===========================+=======+
| Aconitum napellus         | True                      | Aconitum napellus         |       |
+---------------------------+---------------------------+---------------------------+-------+
| Adenium obesum            | True                      | Adenium obesum            |       |
+---------------------------+---------------------------+---------------------------+-------+
| Alpinia purpurata         | True                      | Alpinia purpurata         |       |
+---------------------------+---------------------------+---------------------------+-------+
| Alstroemeria aurea        | True                      | Alstroemeria aurea        |       |
+---------------------------+---------------------------+---------------------------+-------+
| Anemone coronaria         | True                      | Anemone coronaria         |       |
+---------------------------+---------------------------+---------------------------+-------+
| Anemone hupehensis        | True                      | Eriocapitella hupehensis  |       |
+---------------------------+---------------------------+---------------------------+-------+
| Anthurium andraeanum      | True                      | Anthurium andraeanum      |       |
+---------------------------+---------------------------+---------------------------+-------+
```

Figure 3-1: Kiểm tra tên khoa học

3.2.2. Thu thập ảnh theo loài

Sau khi hoàn thiện danh sách tên khoa học, dữ liệu ảnh được thu thập tự động thông qua crawl từ nền tảng **iNaturalist** – một nguồn cung cấp ảnh thực địa có độ tin cậy cao.

Quy trình thu thập bao gồm các bước:

– File CSV tổng hợp chứa trường ten_khoa_hoc được sử dụng làm đầu vào.

```
ten_khoa_hoc
Aconitum napellus
Adenium obesum
Alpinia purpurata
Alstroemeria aurea
Anemone coronaria
Anemone hupehensis
Anthurium andraeanum
Antigonon leptopus
```

Figure 3-2: File CSV lưu tên khoa học

– Với mỗi tên khoa học, chương trình crawler sẽ truy vấn API hoặc trang tìm kiếm của iNaturalist để thu thập danh sách ảnh liên quan. API cơ bản của trang web được gọi qua url như sau:

```
url = (
    "https://api.inaturalist.org/v1/observations"
    f"?taxon_name={quote(species)}"
    "&per_page=200"
    f"&page={page}"
)
```

Figure 3-3: API inaturalist.org

– Ví quá trình thu thập khác lâu nên khuyến khích sử dụng đa luồng để đẩy nhanh tiến độ thu thập (ví dụ dùng 8 luồng):

```
with ThreadPoolExecutor(max_workers=8) as executor:
```

Figure 3-4: Đa luồng thu thập ảnh

– Các ảnh được tải xuống và lưu trữ theo cấu trúc thư mục dạng: /Vietnam_flower/<ten_khoa_hoc>/

Điều này giúp dễ dàng phân loại, kiểm tra chất lượng và huấn luyện mô hình sau này.

Nhờ việc kết hợp giữa danh sách tên khoa học thu thập thủ công và quá trình crawl ảnh tự động, bộ dữ liệu hình ảnh đầu ra đảm bảo tính đa dạng, góp phần nâng cao hiệu quả của mô hình phân loại hoa.

Tổ chức thư mục:

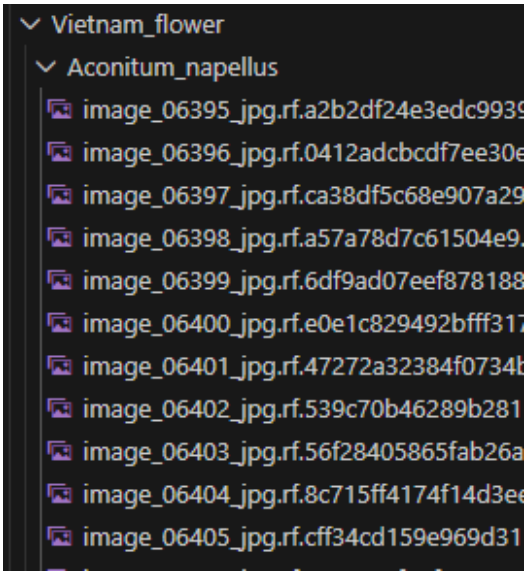


Figure 3-5: Tổ chức thư mục sau khi thu thập

3.3. Tiền xử lý ảnh làm bộ dữ liệu

Sau khi hoàn thành quá trình thu thập ảnh từ nhiều nguồn khác nhau, đặc biệt là nhóm dữ liệu ***Vietnam_flower***, bước tiền xử lý được thực hiện nhằm chuẩn hóa và nâng cao chất lượng dữ liệu trước khi đưa vào mô hình huấn luyện. Do dữ liệu thực tế chứa nhiều nhiễu (noise), ảnh không đúng đối tượng hoặc chất lượng không đồng đều, quá trình tiền xử lý bao gồm hai giai đoạn chính:

3.3.1. Chuẩn hóa kích thước ảnh (Image Resizing)

Ảnh thu thập từ iNaturalist và các nguồn khác có kích thước rất đa dạng, từ vài trăm đến vài nghìn pixel. Điều này gây khó khăn cho mô hình trong việc trích xuất đặc trưng và làm tăng thời gian huấn luyện.

Kiểm tra kích thước ảnh sau khi crawl:

```
Vietnam_flower/Barringtonia_acutangula/32.jpg: 165x147
Vietnam_flower/Eustoma_grandiflorum/311566622_0.jpg: 1024x1024
Vietnam_flower/Bauhinia_purpurea/12.jpg: 426x486
```

Figure 3-6: Kích thước ảnh

Vì vậy, tất cả ảnh được resize về một kích thước chuẩn (224 x 224) và lưu vào ***Dataset***:

```
def resize_image(src_img_path, dst_img_path):
    try:
        img = Image.open(src_img_path)
        img = img.resize((224, 224), Image.LANCZOS)
        img.save(dst_img_path)
    except Exception as e:
        print(f"Lỗi resize {src_img_path}: {e}")
```

Figure 3-7: Resize ảnh

Việc chuẩn hóa kích thước ảnh giúp: Đồng bộ hóa đầu vào cho mô hình, giảm chi phí tính toán, và tăng sự ổn định của quá trình huấn luyện.

Thuật toán resize được thực hiện đồng nhất trên toàn bộ dataset, đồng thời giữ tỉ lệ khung hình khi cần thiết để tránh làm biến dạng hoa.

2. Lọc ảnh thủ công (Manual Cleaning)

Bộ dữ liệu *Vietnam_flower* là dữ liệu thu thập từ môi trường thực tế, do đó tồn tại các lỗi phổ biến như:

- Ảnh không chứa hoa hoặc chứa đối tượng khác.
- Ảnh bị mờ, bị nhiễu hoặc thiếu ánh sáng.
- Ảnh chỉ hiển thị lá, cành mà không rõ phần hoa.
- Ảnh bị che khuất hoặc khó nhận dạng.

Để đảm bảo chất lượng đầu vào, một bước **kiểm tra thủ công** được thực hiện nhằm loại bỏ các ảnh không phù hợp.

Việc lọc thủ công giúp mô hình tránh học phải dữ liệu nhiễu và tăng độ chính xác ở giai đoạn huấn luyện – đánh giá.

Quy trình lọc gồm:

- Duyệt từng thư mục theo tên khoa học.
- Loại bỏ ảnh không đúng loài hoặc không thể nhận dạng.
- Giữ lại những ảnh rõ hoa, đúng chủng loại và đủ thông tin hình thái.

Sau khi lọc thủ công, thì thư mục có các ảnh có độ tương đồng gần như nhau:

Figure 3-8: Ảnh sau khi lọc thủ công

Nhờ hai bước tiền xử lý — **resize ảnh** và **lọc thủ công** — bộ dữ liệu cuối cùng đạt chất lượng cao hơn, giúp mô hình trích xuất đặc trưng chính xác và giảm sai lệch trong quá trình học.

3.4. Phân chia tập dữ liệu

Bộ dữ liệu có kích thước >10.000 ảnh với 113 loài được chia theo tỷ lệ 0.75 / 0.15 / 0.15 tương ứng với ba tập train, validation và test.

Table 3-1: Phân chia tập dữ liệu

Tập dữ liệu	Số lượng	Mục đích sử dụng
Train	7.521	Huấn luyện mô hình (training)
Validation	1.571	Điều chỉnh siêu tham số
Test	1.722	Đánh giá hiệu suất mô hình
Total	>10.000	

Và cấu trúc thư mục sau khi chia như sau:

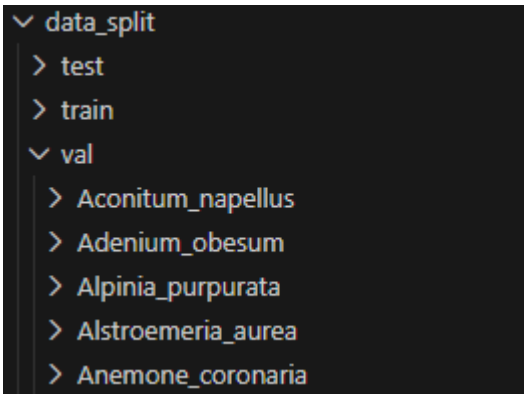


Figure 3-9: Cấu trúc thư mục sau khi chia

3.5. Trực quan hóa dữ liệu

Để có cái nhìn trực quan hơn về dữ liệu, ta có thể thực hiện một số bước khám phá dữ liệu (data exploration) như sau:

3.5.1. Biểu đồ phân phối ảnh theo loài hoa

Để đánh giá mức độ phân bố dữ liệu trong bộ ảnh các loài hoa, trước tiên chúng tôi tiến hành thống kê số lượng ảnh tương ứng với từng loài. Kết quả được biểu diễn dưới dạng biểu đồ cột, giúp quan sát trực quan sự chênh lệch về dữ liệu giữa các lớp.

a) Bảng đồ phân phối cho toàn bộ tập dữ liệu

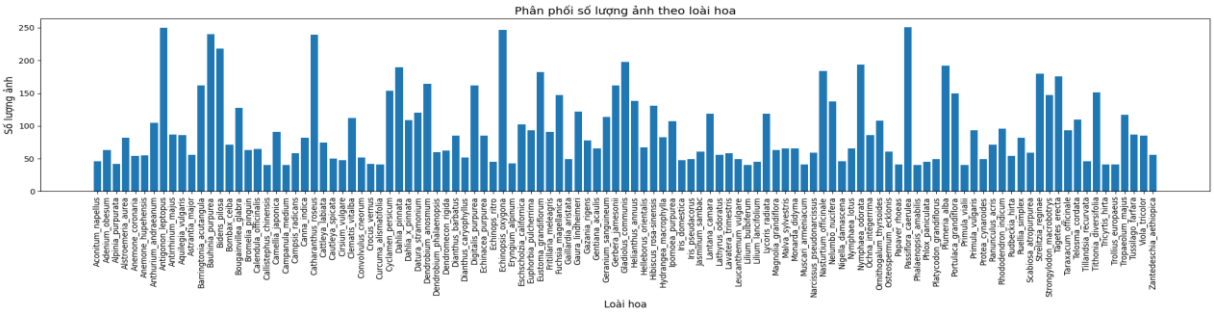


Figure 3-10: Phân phối dữ liệu theo loài

Nhận xét:

- Phân bố không đồng đều: Số lượng ảnh giữa các loài hoa chênh lệch khá lớn. Một số loài có trên 200 ảnh, trong khi nhiều loài khác chỉ có khoảng 40–80 ảnh.
- Có một số loài nổi bật với số lượng ảnh rất cao, các cột vượt trội (cao nhất đồ thị) nằm ở mức ~230–250 ảnh.

- Có nhiều loài ảnh khá ít: Một số loài chỉ khoảng 40–60 ảnh. Điều này có thể gây mất cân bằng dữ liệu khi huấn luyện mô hình phân loại.
- Gần như không có loài nào bị thiếu ảnh hoàn toàn, dataset tương đối đầy đủ, mỗi loài đều có dữ liệu.
- 5. Số lượng loài hoa rất lớn
- Biểu đồ chứa hàng trăm loài → dataset đa dạng, phù hợp cho mô hình phân loại số lớp lớn (multi-class classification).
- Mặc dù có sự chênh lệch, nhưng hầu hết các loài nằm quanh mức 60–160 ảnh, tức là dataset có xu hướng phân bố trung bình khá ổn.

b) Bảng đồ phân phối cho tập train

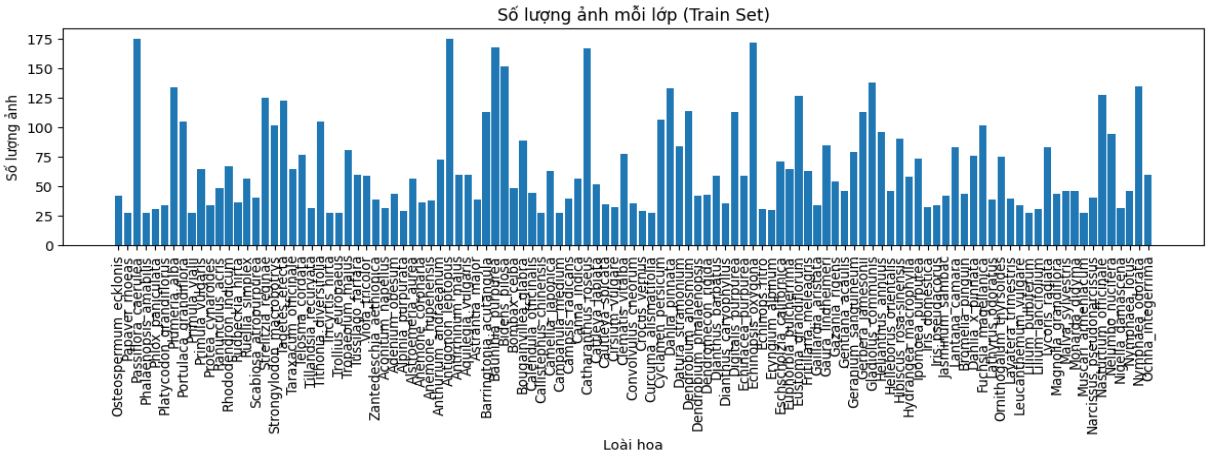


Figure 3-11: Phân phối dữ liệu theo loài trên tập train

Nhận xét:

- Phân bố số lượng ảnh trong Train Set nhìn chung giống với toàn bộ dữ liệu, giữ đúng xu hướng các lớp nhiều/ít ảnh.
- Quy mô Train Set nhỏ hơn nên số lượng ảnh mỗi lớp giảm, nhưng tỷ lệ phân bố vẫn ổn định và không tạo ra mất cân bằng mới.
- Mất cân bằng giữa các lớp vẫn còn như trong toàn tập, vì vậy khi huấn luyện cần xem xét data augmentation hoặc class weighting.

3.5.2. Trực quan hóa so sánh các tập dữ liệu

c) Số lượng ảnh

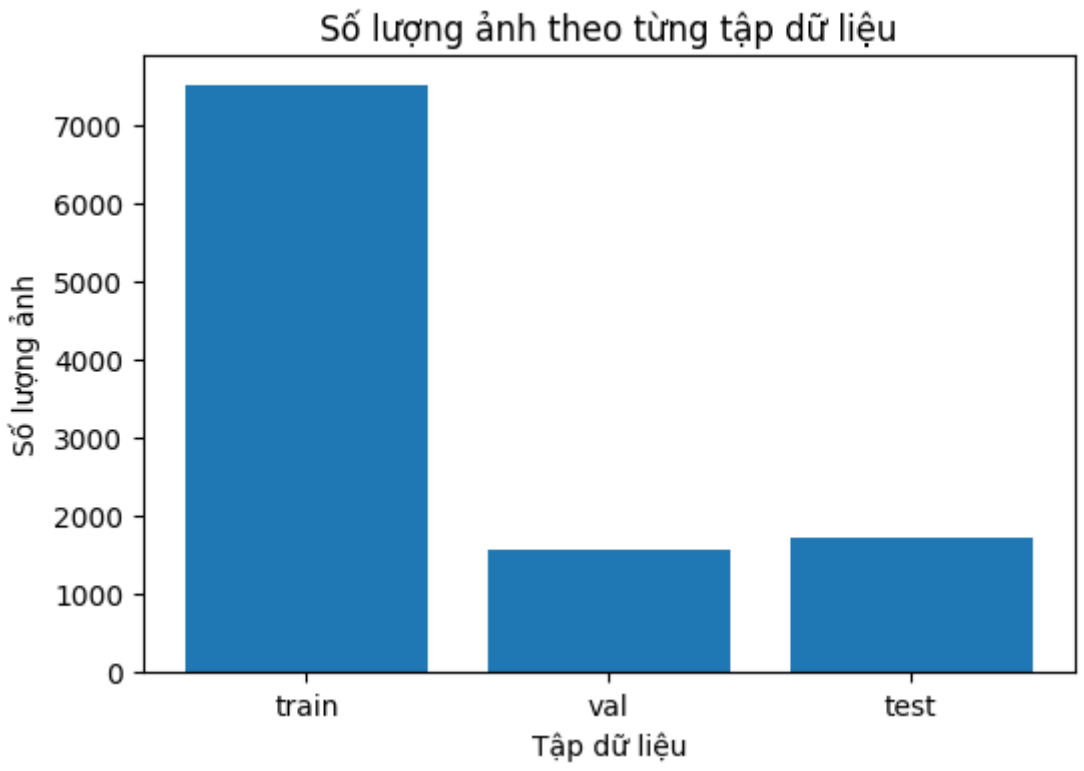


Figure 3-12: Số lượng ảnh từng tập dữ liệu

Bộ dữ liệu được chia thành 7.521 ảnh Train, 1.571 ảnh Validation và 1.722 ảnh Test.

Train Set chiếm phần lớn giúp mô hình có đủ dữ liệu để học sâu các đặc trưng của từng loài hoa, đồng thời giảm nguy cơ học thiếu hoặc không ổn định.

Validation Set và Test Set đều có quy mô đủ lớn, đảm bảo việc tinh chỉnh siêu tham số, theo dõi overfitting và đánh giá cuối cùng của mô hình được khách quan, tin cậy và phản ánh đúng hiệu suất trên dữ liệu mới.

d) Tỷ lệ phân chia

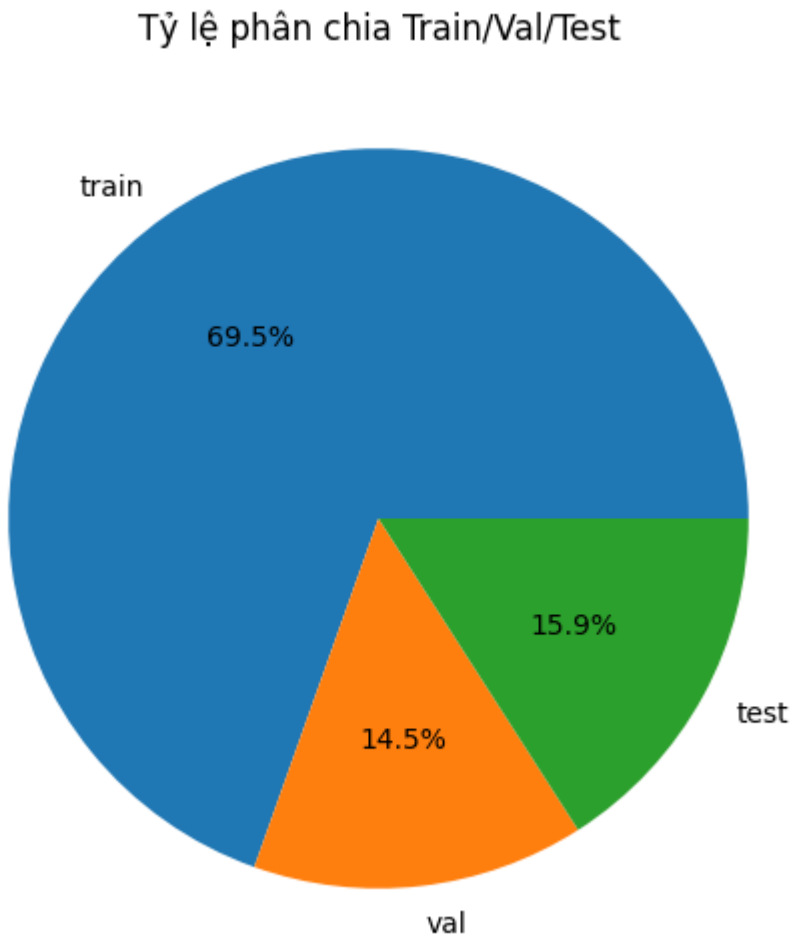


Figure 3-13: Tỷ lệ phân chia thực tế

Tỷ lệ 69.5% cho Train, 14.5% cho Validation và 15.9% cho Test nhìn chung rất hợp lý, gần với chuẩn thường dùng (70% – 15% – 15%), đảm bảo dữ liệu được phân bổ cân đối giữa huấn luyện và đánh giá.

Train Set chiếm gần 70% giúp mô hình có đủ mẫu để học đặc trưng, trong khi việc dành ra 14.5% cho Validation hỗ trợ theo dõi overfitting và tối ưu siêu tham số một cách hiệu quả.

Test Set chiếm khoảng 15.9% mang lại độ tin cậy cao khi đánh giá cuối cùng, đảm bảo mô hình được kiểm nghiệm trên một phần dữ liệu độc lập và đủ lớn để phản ánh đúng hiệu suất thực tế.

CHƯƠNG 4: XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH

4.1. Các kỹ thuật huấn luyện mô hình

4.1.1. Tăng cường dữ liệu

Trong quá trình huấn luyện, tôi sử dụng *ImageDataGenerator* của Keras để tăng cường dữ liệu trực tiếp trên ảnh huấn luyện. Cụ thể, tôi áp dụng các phép biến đổi sau:

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Figure 4-1: Phương pháp ImageDataGenerator

Các biến đổi này được dùng cho tập train để làm cho ảnh đầu vào đa dạng hơn, gồm xoay ±30°, tịnh tiến ngang/dọc 20%, biến dạng shear 20%, zoom ±20%, lật ngang và chuẩn hóa pixel.

Ngoài các phép biến đổi cơ bản, dùng hàm **Mixup** để trộn hai ảnh và hai nhãn với nhau trước khi đưa vào mô hình.

```
def mixup(self, image1, image2, label1, label2):

    if self.mixup_alpha > 0:
        lam = np.random.beta(self.mixup_alpha, self.mixup_alpha)
    else:
        lam = 1.0

    # Mix images
    mixed_image = lam * image1 + (1 - lam) * image2

    # Mix labels
    mixed_label = lam * label1 + (1 - lam) * label2

    return mixed_image, mixed_label
```

Figure 4-2: Phương pháp MixUp

Trong quá trình tạo batch huấn luyện, mỗi cặp ảnh được chọn ngẫu nhiên để áp dụng Mixup với hệ số λ lấy từ phân phối Beta. Ảnh sau khi trộn được sử dụng trực tiếp để train mô hình.

4.1.2. Class Weights

Phương pháp **Class Weight** giúp mô hình giảm thiên lệch khi dữ liệu bị mất cân bằng giữa các lớp.

```
class_to_idx = {cls: idx for idx, cls in enumerate(flower_classes)}
idx_to_class = {idx: cls for cls, idx in class_to_idx.items()}

class_weights_array = compute_class_weight(
    class_weight='balanced',
    classes=np.arange(len(flower_classes)),
    y=[class_to_idx[flower] for flower in flower_classes
       for _ in range(flower_counts[flower])]
)

class_weights = {i: weight for i, weight in enumerate(class_weights_array)}
```

Figure 4-3: Phương pháp Class Weight

- Tăng trọng số lỗi cho lớp ít dữ liệu → mô hình sẽ phạt nặng hơn khi dự đoán sai các lớp hiếm
- Giảm trọng số lỗi cho lớp nhiều dữ liệu → tránh để lớp lớn chi phối hoàn toàn quá trình học

4.1.3. Focal loss

Hàm loss tùy chỉnh Focal Loss nhằm giúp mô hình tập trung nhiều hơn vào các mẫu khó dự đoán.

```
def call(self, y_true, y_pred):
    # Clip predictions để tránh log(0)
    epsilon = K.epsilon()
    y_pred = K.clip(y_pred, epsilon, 1.0 - epsilon)

    # Cross entropy
    ce = -y_true * K.log(y_pred)

    # Focal term: (1 - p_t)^gamma
    focal_weight = K.pow(1 - y_pred, self.gamma)

    # Focal loss
    focal_loss = self.alpha * focal_weight * ce

    return K.sum(focal_loss, axis=-1)
```

Figure 4-4: Phương pháp Focal Loss

- Clip giá trị dự đoán: Tránh tình huống log (0), giúp quá trình tính loss ổn định hơn.
- Tính Cross-Entropy cơ bản
- Tính trọng số Focal (focal term)
  + Nếu mô hình dự đoán đúng và tự tin (y_pred ≈ 1) → trọng số gần 0
  + Nếu mô hình dự đoán sai hoặc không tự tin → trọng số lớn → Tập trung học vào mẫu khó, giảm ảnh hưởng của mẫu dễ.
- Áp dụng hệ số cân bằng alpha
  + alpha cân bằng giữa các lớp (thường dùng khi dữ liệu lệch lớp)
  + Kết hợp cả cross-entropy và trọng số khó–dễ.

4.1.4. ReduceLROnPlateau

Callback **ReduceLROnPlateau** để điều chỉnh tốc độ học (learning rate) tự động khi mô hình không còn cải thiện. Đây là phần code được áp dụng:

```
# Reduce learning rate on plateau
ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=5,
    min_lr=1e-7,
    verbose=1
),
```

Figure 4-5: Phương pháp ReduceLROnPlateau

- monitor: Theo dõi giá trị loss trên tập validation.
- patience: Nếu val_loss không giảm sau 5 epoch liên tiếp, callback sẽ tự kích hoạt.
- factor: Khi kích hoạt, learning rate sẽ được nhân với 0.2 → giảm xuống còn 20% so với trước đó.
- min_lr: Learning rate sẽ không giảm thấp hơn mức này để tránh mô hình học quá chậm.
- verbose: In ra thông báo mỗi khi learning rate được giảm.

4.2. Mô hình EfficientNetB0

4.2.1. Kiến trúc mô hình

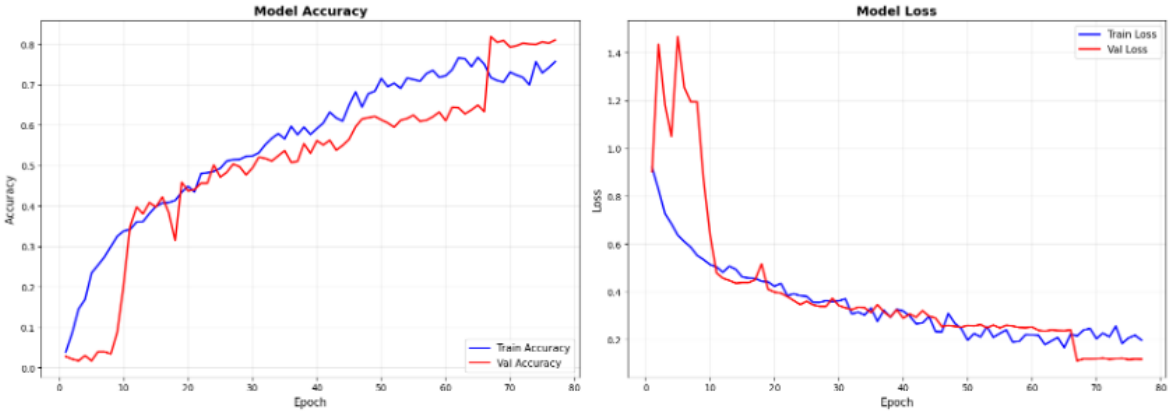
```
INPUT_SHAPE = (224, 224, 3)

# Base model
base_model = EfficientNetB0(
    weights=None,   # nếu muốn dùng pretrained thì đổi thành 'imagenet'
    include_top=False,
    input_shape=INPUT_SHAPE
)
base_model.trainable = True

x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.3)(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.3)(x)
predictions = layers.Dense(NUM_CLASSES, activation='softmax')(x)

model = models.Model(inputs=base_model.input, outputs=predictions)

model.summary()
```

Figure 4-6: Build model EfficientNetB0

4.2.2. Siêu tham số

- Learning rate: 1e-3, được điều chỉnh tự động bằng ReduceLROnPlateau, giảm xuống 0.5 lần nếu val_loss không cải thiện.
- Batch size: 64, phù hợp với kích thước ảnh và tài nguyên GPU đang sử dụng.
- Số epoch: 300, sử dụng EarlyStopping để dừng sớm nếu mô hình không còn cải thiện.
- Optimizer: Adam, hội tụ nhanh, ổn định.
- ReduceLROnPlateau:
  + Monitor = ‘val_loss’
  + factor=0.5
  + patience=5
  + min_lr=1e-7
  + verbose=1

4.2.3. Quá trình huấn luyện




```
============================================================
Training Summary:
============================================================
Total epochs trained: 77

Best validation accuracy: 0.8180
  at epoch: 67
Train accuracy at best epoch: 0.7172
Val loss at best epoch: 0.1108
============================================================
```

Figure 4-7: Quá trình huấn luyện EfficientNetB0

Nhận xét:

- Train accuracy và validation accuracy đều tăng ổn định theo thời gian, cho thấy mô hình học tốt.
- Hai đường accuracy nằm khá gần nhau, chứng tỏ mô hình không bị overfitting.
- Train loss và val loss giảm mạnh trong giai đoạn đầu, thể hiện mô hình học nhanh và hiệu quả.
- Sau khoảng 50–60 epoch, loss ổn định và hội tụ, chỉ dao động nhẹ.
- Nhìn chung, mô hình đạt trạng thái ổn định và đã học được đặc trưng dữ liệu một cách hiệu quả.

4.3. Mô hình Resnet

4.3.1. Kiến trúc mô hình

```python
# Initialize ResNet50 from scratch (no pretrained weights)
model = models.resnet50(weights=None)
print(f"Initialized ResNet50 from scratch")

# Apply weight initialization
def init_weights(m):
    if isinstance(m, nn.Conv2d):
        if init_method == "kaiming":
            nn.init.kaiming_normal_(m.weight, mode="fan_out", nonlinearity="relu")
        elif init_method == "xavier":
            nn.init.xavier_normal_(m.weight)
        if m.bias is not None:
            nn.init.constant_(m.bias, 0)
    elif isinstance(m, nn.BatchNorm2d):
        nn.init.constant_(m.weight, 1)
        nn.init.constant_(m.bias, 0)
    elif isinstance(m, nn.Linear):
        nn.init.normal_(m.weight, 0, 0.01)
        if m.bias is not None:
            nn.init.constant_(m.bias, 0)

model.apply(init_weights)
print(f"Applied {init_method} initialization to all layers")

# Get the number of features in the final FC layer
num_features = model.fc.in_features
print(f"Original FC layer: in_features={num_features}")

# Replace final FC layer with custom classifier
model.fc = nn.Sequential(
    nn.Dropout(p=dropout_rate), nn.Linear(num_features, num_classes)
)

# Initialize new classifier
nn.init.normal_(model.fc[1].weight, 0, 0.01)
nn.init.constant_(model.fc[1].bias, 0)
```

Figure 4-8: Build model ResNet50

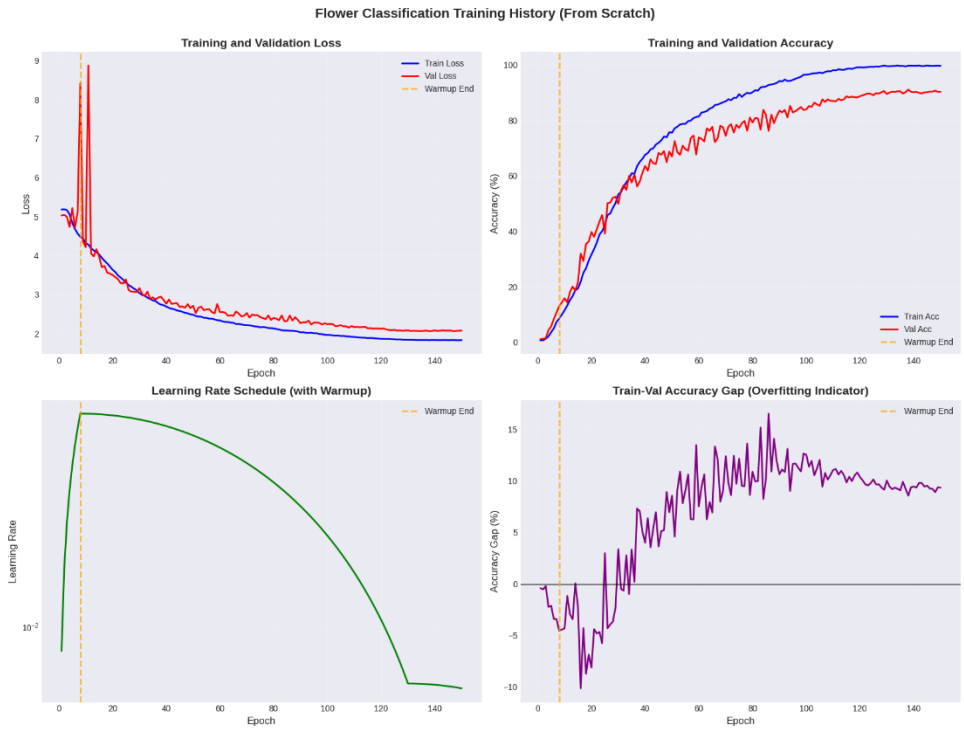
4.3.2. Siêu tham số

- Learning Rate 0.08: LR cao giúp mô hình học nhanh giai đoạn đầu.
- Scheduler: Cosine giảm dần mềm mại, tránh kẹt tại cực tiểu cục bộ, giữ học chậm mà ổn định cuối quá trình
- Min LR: 1e-6
- Warmup Epochs: 8 giảm sốc gradient khi khởi đầu với LR lớn
- Epoch: 150 đủ dài cho mô hình hội tụ với kiến trúc sâu.
- Optimizer: SGD tối ưu truyền thống, ổn định, khả năng generalize tốt hơn Adam trong các bài toán thị giác.

– Label smoothing: 0.2 giảm overfitting, tránh mô hình quá tự tin với 1 nhãn.

– Early Stopping: 20 tiết kiệm tài nguyên, dừng khi mô hình không còn cải thiện, tránh overfitting cuối giai đoạn học.

4.3.3. Quá trình huấn luyện




```
=======
Best validation accuracy: 91.06%
Total training epochs: 150
==================================
```

Figure 4-9: Quá trình huấn luyện ResNet50

Xu hướng chung: Cả train loss (xanh) và val loss (đỏ) giảm liên tục từ epoch đầu đến cuối, chứng tỏ mô hình học tốt và dần hội tụ.

Giai đoạn warmup (khoảng 8 epoch đầu): Loss ban đầu dao động lớn, đặc biệt val loss có spike, điều này là bình thường khi LR đang tăng dần (warmup) để tránh sốc gradient.

Sau warmup: Loss giảm mượt và ổn định, không có hiện tượng plateau sớm hay loss tăng trở lại quá nhiều, cho thấy LR cosine annealing hoạt động hiệu quả.

Tăng đều: Train acc tăng từ ~0% lên ~100%, val acc tăng từ ~0% lên ~90%, cho thấy mô hình học đúng xu hướng nhưng vẫn có khoảng cách giữa train và val.

Khoảng cách train-val: Khoảng 10% là khá chấp nhận được với bài toán phân loại nhiều lớp, không quá lo ngại overfitting nghiêm trọng.

4.4. Mô hình MobileNet

4.4.1. Kiến trúc mô hình

```
# Load pre-trained MobileNet
base_model = MobileNet(
    weights=None,
    include_top=False,
    input_shape=input_shape,
    alpha=1.0,  # Width multiplier
    depth_multiplier=1
)

# Unfreeze ALL layers for fine-tuning
base_model.trainable = True

# Add custom classification head
model = keras.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.5),
    layers.Dense(512, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(num_classes, activation='softmax', name='predictions')
])
```

Figure 4-10: Build model MobileNet

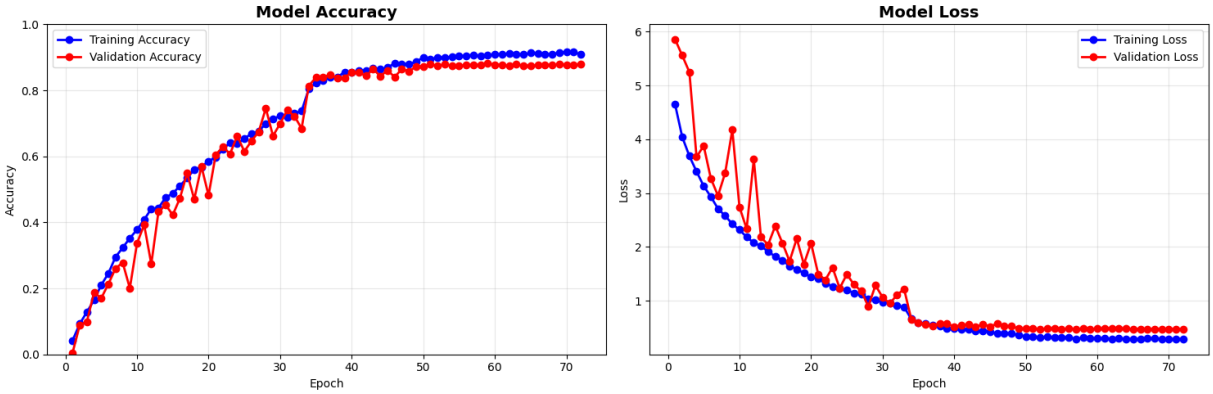
4.4.2. Siêu tham số

- Learning rate: 1e-3, được điều chỉnh tự động bằng ReduceLROnPlateau, giảm xuống 0.2 lần nếu val_loss không cải thiện.
- Batch size: 16, phù hợp với kích thước ảnh và tài nguyên GPU đang sử dụng.
- Số epoch: 300, sử dụng EarlyStopping để dừng sớm nếu mô hình không còn cải thiện.
- Optimizer: Adam, hội tụ nhanh, ổn định.
- ReduceLROnPlateau**:**
  + Monitor = ‘val_loss’
  + factor=0.2
  + patience=5
  + min_lr=1e-7

+ verbose=1

4.4.3. Quá trình huấn luyện

Khi không áp dụng tăng cường dữ liệu bằng phương pháp MixUp và kĩ thuật Focal Loss:



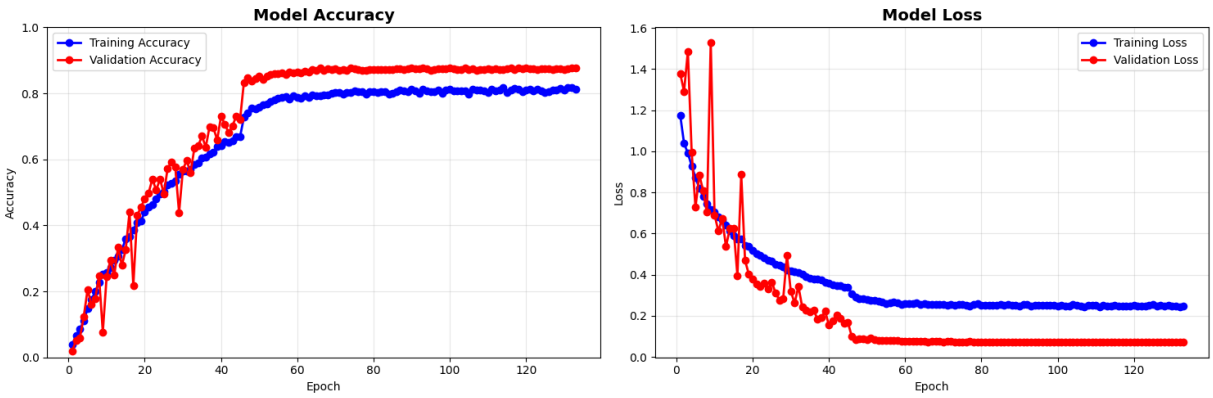

```
Training Accuracy: 0.9088
Validation Accuracy: 0.8791
Training Loss: 0.2965
Validation Loss: 0.4766
```

Figure 4-11: Quá trình huấn luyện MobileNet không dùng phương pháp

Nhận xét:

- Sau khoảng epoch 35–40, độ chính xác đạt khoảng 85–90% và gần như không tăng thêm nhiều nữa → mô hình đã gần đạt ngưỡng tối ưu.
- Đường Validation gần sát với Training → mô hình không bị overfitting quá mạnh.
- Validation Loss dao động mạnh ở giai đoạn đầu (điều này bình thường vì mô hình chưa ổn định).
- Sau khoảng epoch 30, Validation Loss giảm ổn định và tiến sát training loss.
- Đến cuối Training (~epoch 70), loss gần hội tụ → mô hình học tốt và không bị underfitting.

Khi áp dụng tăng cường dữ liệu bằng phương pháp MixUp và kĩ thuật Focal Loss:




```
Training Accuracy: 0.8112
Validation Accuracy: 0.8759
Training Loss: 0.2477
Validation Loss: 0.0712
```

Figure 4-12: Quá trình huấn luyện MobileNet có dùng phương pháp

Nhận xét:

- Validation Loss và Train Loss rất tốt, thấp hơn nhiều.
- Validation Accuracy cao hơn Training Accuracy, do mô do mô hình áp dụng phương pháp khiến tập train khó học hơn nhưng có thể giúp tập dữ liệu Val tổng quát tốt hơn

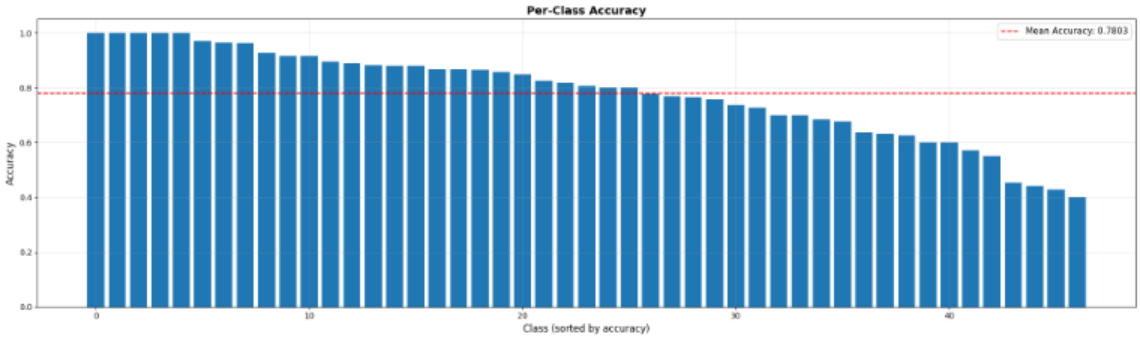
CHƯƠNG 5: ĐÁNH GIÁ VÀ SO SÁNH MÔ HÌNH

5.1. Kết quả trên tập test

5.1.1. Mô hình EfficientNetB0

```
==================================================
Test Results:
==================================================
Test Loss: 0.1144
Test Accuracy: 0.8040 (80.40%)
==================================================
```

Figure 5-1: Kết quả trên tập Test (EfficientNetB0)




```
Top 5 lớp có độ chính xác cao nhất:
                      Class  Accuracy  Samples
28           Lycoris_radiata       1.0       19
29         Magnolia_denudata       1.0        5
40        Strelitzia_reginae       1.0       28
41  Strongylodon_macrobotrys       1.0       23
16       Fuchsia_magellanica       1.0       23

Top 5 lớp có độ chính xác thấp nhất:
                     Class  Accuracy  Samples
34   Phalaenopsis_amabilis  0.550000       20
13   Dianthus_caryophyllus  0.454545       22
39          Rosa_chinensis  0.440000       25
33    Paeonia_suffruticosa  0.428571       14
26  Lavandula_angustifolia  0.400000        5
```

Figure 5-2: Phân phối Accuracy từng loài trên tập Test (EfficientNetB0)

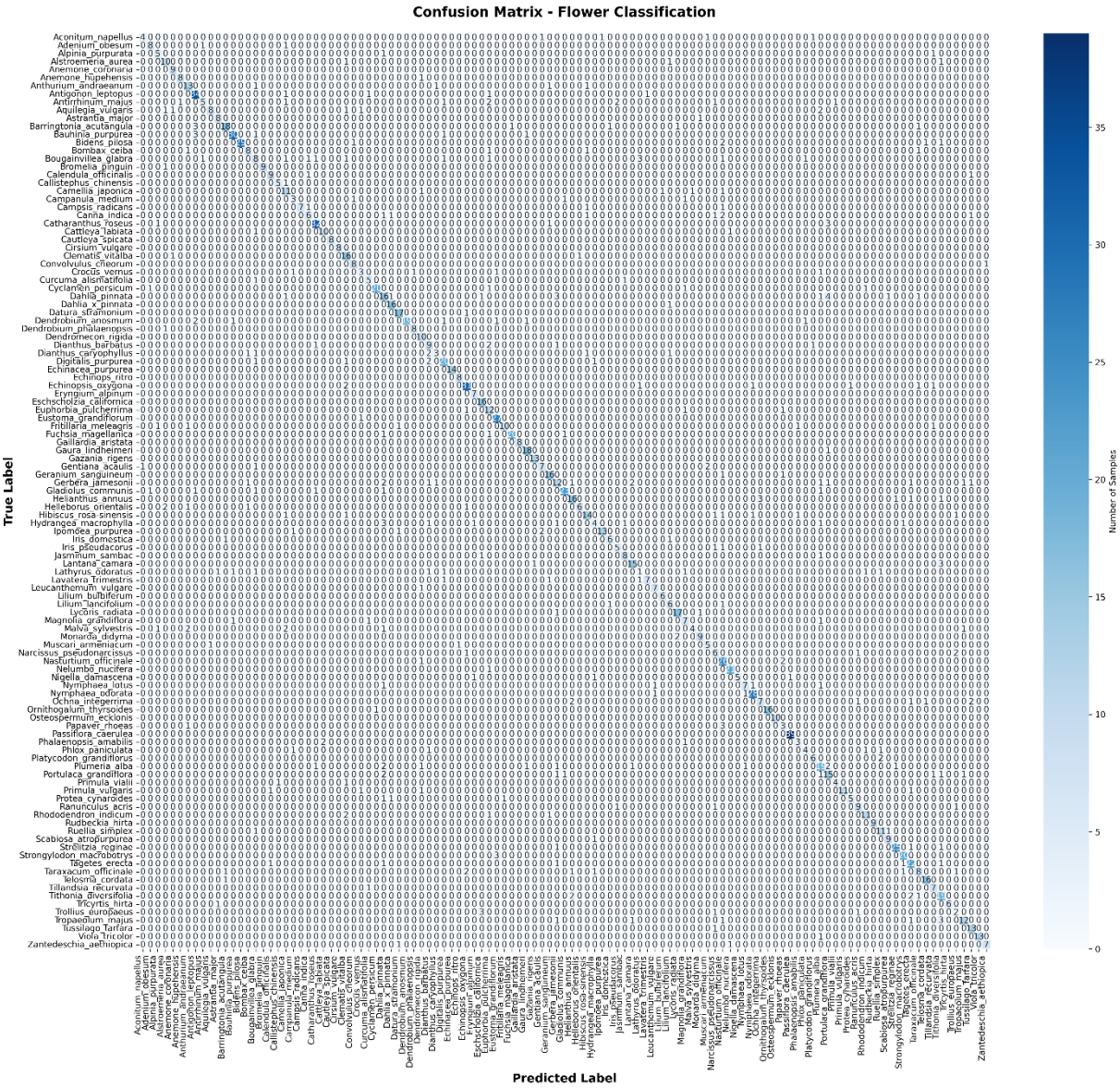


Figure 5-3: Confusion Matrix (EfficientNetB0)

5.1.2. Mô hình Resnet

```
================================
TEST SET EVALUATION
================================
Test Accuracy (Top-1): 91.00%
Test Accuracy (Top-3): 96.49%
Test Accuracy (Top-5): 97.72%
Total test samples: 1623
================================
```

Figure 5-4: Kết quả trên tập Test (ResNet50)

```
Per-Class Accuracy:
============================================================
Class Name                      Accuracy        Samples
============================================================

Top 5 Best Performing Classes:
Aconitum_napellus               100.00%         7
Astrantia_major                 100.00%         8
Strongylodon_macrobotrys        100.00%         22
Strelitzia_reginae              100.00%         27
Scabiosa_atropurpurea           100.00%         9

Top 5 Worst Performing Classes:
Lathyrus_odoratus                37.50%         8
Dianthus_barbatus                61.54%         13
Dianthus_caryophyllus            62.50%         8
Campanula_medium                 66.67%         6
Phlox_paniculata                 66.67%         6
============================================================
```

Figure 5-5: Top 5 class tốt nhất và tệ nhất (Resnet 50)

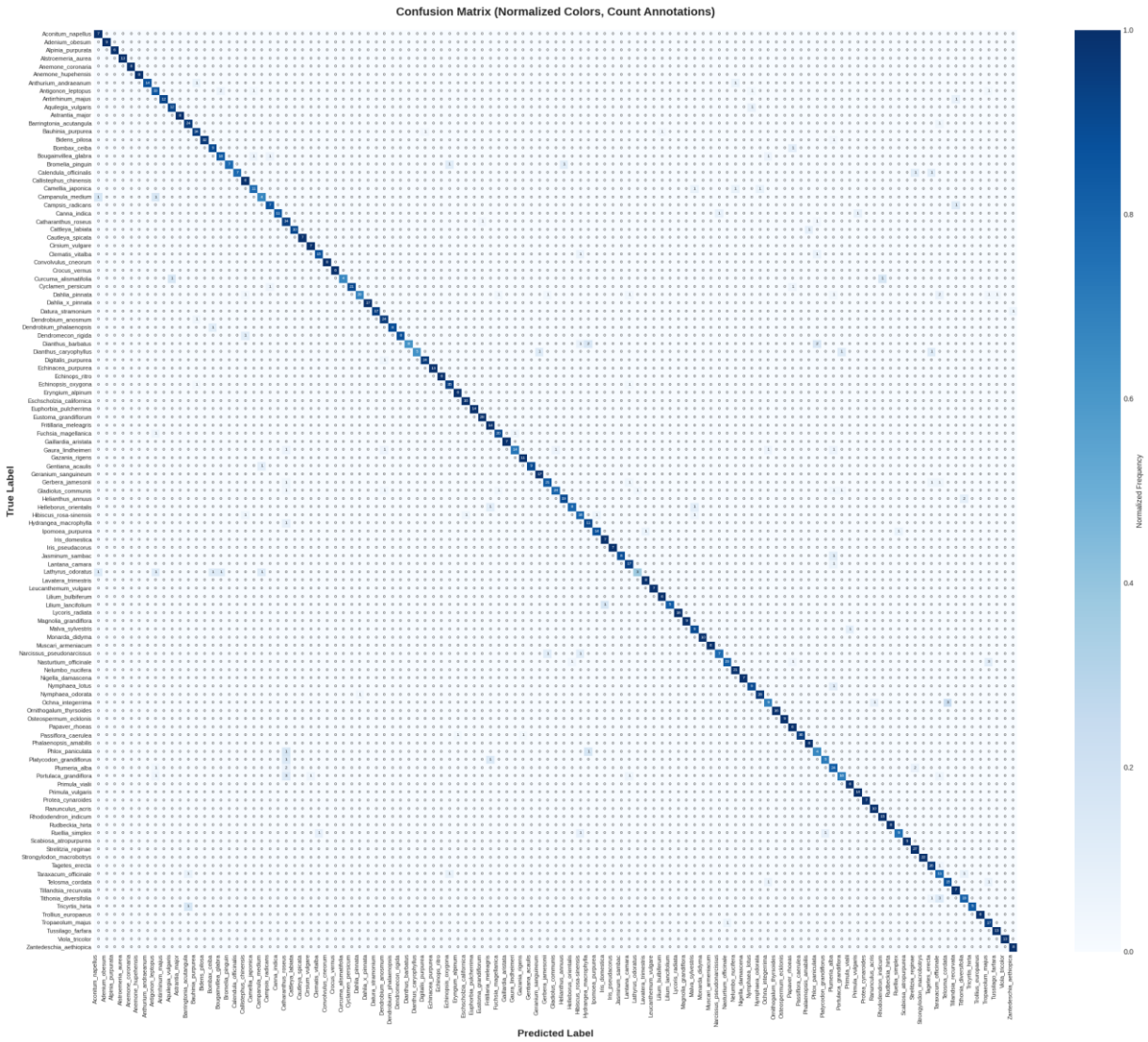


Figure 5-6: Confusion Matrix (ResNet50)

5.1.3. Mô hình MobileNet

a) Khi không dùng phương pháp MixUp và Focal Loss (1)

```
Top-1-Accuracy: 0.8792 (87.92%)
Top-3-Accuracy: 0.9535 (95.35%)
Top-5-Accuracy: 0.9715 (97.15%)
Test Loss: 0.4847
```

Figure 5-7: Kết quả trên tập Test (MobileNet) (1)

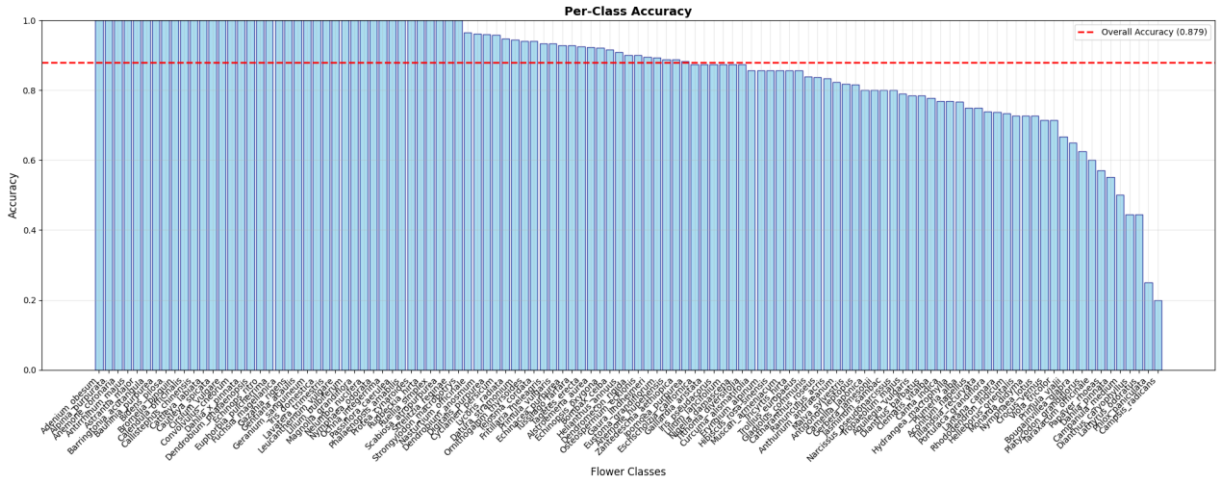


Figure 5-8: Phân phối Accuracy từng loài trên tập Test (MobileNet) (1)

b) Khi dùng phương pháp MixUp và Focal Loss (2)

```
Top-1-Accuracy: 0.8513 (85.13%)
Top-3-Accuracy: 0.9564 (95.64%)
Top-5-Accuracy: 0.9756 (97.56%)
Test Loss: 0.0859
```

Figure 5-9: Kết quả trên tập Test (MobileNet) (2)

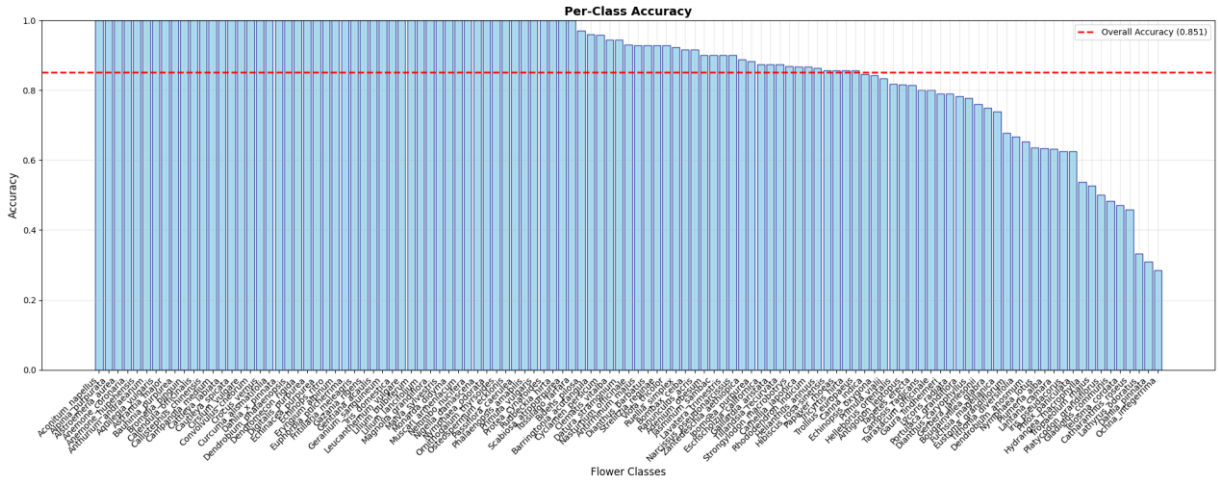


Figure 5-10: Phân phối Accuracy từng loài trên tập Test (MobileNet) (2)

c) So sánh 2 phương pháp

Table 5-1: Bảng so sánh 2 phương pháp MobileNet

(1)	(2)
Overall Accuracy: 0.8792 (87.92%) Mean Per-Class Accuracy: 0.8674 (86.74%) Std Per-Class Accuracy: 0.1567 Min Per-Class Accuracy: 0.2000 (20.00%) Max Per-Class Accuracy: 1.0000 (100.00%)	Overall Accuracy: 0.8513 (85.13%) Mean Per-Class Accuracy: 0.8769 (87.69%) Std Per-Class Accuracy: 0.1686 Min Per-Class Accuracy: 0.2857 (28.57%) Max Per-Class Accuracy: 1.0000 (100.00%)

Nhận xét:

- Mô hình không còn thiên lệch vào các lớp nhiều mẫu → học cân bằng hơn
- Hiệu suất trung bình trên từng lớp tăng → mô hình hiểu dữ liệu đa dạng hơn
- Lớp có độ chính xác thấp nhất được cải thiện rõ rệt
- Tính công bằng giữa các lớp tốt hơn, giảm nguy cơ mô hình bị mù lớp hiếm
- Đánh đổi hợp lý: giảm nhẹ Overall Accuracy nhưng tăng độ tin cậy và độ bao phủ của mô hình

5.2. So sánh mô hình

Table 5-2: So sánh mô hình

	EfficientNetB0	**ResNet50**	**MobileNet**
Validation Accuracy	81%	91,06%	87,69%
Test Accuracy	81%	91%	85,13%
F1-Score	81%	91%	85%
Tham số huấn luyện	4,8 tr	23,7 tr	3,8 tr
Best Epoch	67 epoch	137 epoch	133 epoch
Thời gian train mỗi epoch	100s	60s	55s

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Trong đề tài nhận dạng hoa, nhóm/đề tài đã tiến hành huấn luyện và đánh giá hiệu quả của nhiều kiến trúc CNN khác nhau kết hợp với các kỹ thuật tối ưu hóa quá trình học như tăng cường dữ liệu (Data Augmentation), Focal Loss, Class Weight để xử lý mất cân bằng dữ liệu, và tinh chỉnh các siêu tham số.

Các kết quả thí nghiệm cho thấy mô hình có khả năng phân loại hoa với độ chính xác cao trên tập kiểm thử. Việc áp dụng Data Augmentation giúp cải thiện khả năng tổng quát hóa, giảm hiện tượng overfitting. Trong khi đó, Focal Loss và Class Weight góp phần nâng cao độ chính xác đối với các lớp dữ liệu ít, đảm bảo mô hình không bị thiên lệch dự đoán.

Kết quả cuối cùng đạt được độ chính xác validation khoảng ~87–90%, với Validation Loss thấp và ổn định, chứng minh mô hình đã học tốt đặc trưng hình ảnh và đạt được hiệu quả cao trong bài toán phân loại đa lớp.

6.2. Hướng phát triển

Trong tương lai, đề tài có thể được mở rộng và cải thiện theo các hướng sau:

- Tối ưu mô hình
  - Fine-tuning sâu hơn trên backbone pretrained để tăng thêm độ chính xác.
  - Thử nghiệm các kiến trúc hiện đại hơn như EfficientNetV2, ConvNeXt, Vision Transformer (ViT).
- Mở rộng dữ liệu
  - Thu thập thêm dữ liệu từ nhiều nguồn và môi trường chụp ảnh khác nhau.
  - Áp dụng kỹ thuật Semi-supervised Learning hoặc Synthetic Image Generation (GAN) để bổ sung dữ liệu.
- Cải thiện độ tin cậy khi sử dụng thực tế
  - Sử dụng thêm Attention Mechanism giúp mô hình tập trung vào vùng hoa đặc trưng. Hỗ trợ phân loại nhiều loài hoa gần giống nhau.
  - Kết hợp thông tin siêu dữ liệu như màu sắc chủ đạo, hình dạng cánh, ...

TÀI LIỆU THAM KHẢO

Nguồn dữ liệu về tên khoa học:

[1] https://hoavietnam.vn/

[2] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/categories.html

[3] https://www.1800flowers.com/articles/flower-facts/flower-types

Nguồn dữ liệu để cào ảnh từ tên khoa học :

[4] https://www.inaturalist.org/

Tài liệu học thuật

[5] https://viblo.asia/p/cnn-architecture-series-1-mobilenets-mo-hinh-gon-nhe-cho-mobile-applications-1VgZvJV1ZAw

[6] MobileNetV3: https://arxiv.org/pdf/1905.02244

[7] EffcienetB0: https://arxiv.org/pdf/1905.11946#page=2