**TRƯỜNG ĐẠI HỌC VINH**

**VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

**A blue and white logo with a map and a globe

AI-generated content may be incorrect.**

**TIỂU LUẬN HỌC MÁY**

**PHÂN LOẠI HÌNH ẢNH HOA**

|  |  |
| --- | --- |
| GVHD: | PGS TS. HOÀNG HỮU VIỆT |
| Sinh viên: | Phan Huy Hoàng, 215748020110431(NT)  Đinh Đức Mạnh, 215748020110421 |

**Nghệ An, 6/202****5**

**LỜI MỞ ĐẦU**

Trong kỷ nguyên phát triển mạnh mẽ của Công nghệ thông tin và Trí tuệ nhân tạo, học máy và đặc biệt là học sâu, đã trở thành những công cụ mạnh mẽ, mang lại đột phá trong nhiều lĩnh vực. Thị giác máy tính, một ứng dụng nổi bật của học sâu, cho phép máy tính "nhìn" và "hiểu" hình ảnh, mở ra nhiều ứng dụng thực tiễn quan trọng.

Tiểu luận này tập trung vào bài toán phân loại hình ảnh hoa, một vấn đề cơ bản trong thị giác máy tính. Chúng em đã nghiên cứu, xây dựng và đánh giá hai mô hình học sâu: một Mạng Nơ-ron Tích chập đơn giản (CNN) tự thiết kế và một mô hình ResNet18 sử dụng kỹ thuật học chuyển giao. Mục tiêu là phân loại chính xác các loài hoa từ bộ dữ liệu Oxford 102 Flowers.

Trong quá trình thực hiện, nhóm chúng em đã nỗ lực vận dụng kiến thức lý thuyết vào cài đặt thực tế. Tuy nhiên, do kiến thức và kinh nghiệm còn hạn chế, sản phẩm không thể tránh khỏi những thiếu sót. Vì vậy, chúng em kính mong nhận được những đóng góp quý báu từ các thầy cô để tiểu luận được hoàn thiện và có giá trị hơn.

Cuối cùng, chúng em xin chân thành cảm ơn PGS. TS. Hoàng Hữu Việt và các thầy cô trong Khoa công nghệ thông tin đã hỗ trợ chúng em trong quá trình thực hiện đề tài này.

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 2](#_Toc200398075)

[**CHƯƠNG 1. GIỚI THIỆU BÀI TOÁN** 4](#_Toc200398076)

[**1.1. Mô tả bài toán** 6](#_Toc200398077)

[1.1.1. Tổng quan bài toán phân loại hình ảnh 6](#_Toc200398078)

[1.1.2. Bài toán phân loại hoa 6](#_Toc200398079)

[1.1.3. Mục tiêu nghiên cứu 7](#_Toc200398080)

[1.1.4. Ứng dụng thực tế của bài toán phân loại hoa 7](#_Toc200398081)

[**1.2. Bộ dữ liệu hình ảnh hoa** 8](#_Toc200398082)

[**1.3. Mô tả dữ liệu** 9](#_Toc200398083)

[**1.4. Trực quan dữ liệu** 10](#_Toc200398084)

[**CHƯƠNG 2. XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH** 14](#_Toc200398085)

[**2.1. Mô tả kiến trúc của các mô hình học sâu** 14](#_Toc200398086)

[2.1.1. Mô hình SimpleCNN 14](#_Toc200398087)

[2.1.2. Mô hình ResNet18 21](#_Toc200398088)

[**2.2. Cài đặt mô hình bằng Python** 24](#_Toc200398089)

[2.2.1. Cài đặt mô hình SimpleCNN 25](#_Toc200398090)

[2.2.2. Cài đặt mô hình ResNet18 29](#_Toc200398091)

[**2.3. Đánh giá mô hình CNN và ResNet18** 35](#_Toc200398092)

[2.3.1. Đánh giá mô hình CNN 35](#_Toc200398093)

[2.3.2. Đánh giá mô hình ResNet18 43](#_Toc200398094)

[**2.4. So sánh và giải thích kết quả của mỗi mô hình** 51](#_Toc200398095)

[**CHƯƠNG 3. KẾT LUẬN** 54](#_Toc200398096)

[**3.1. Tóm tắt kết quả đạt được** 54](#_Toc200398097)

[**3.2. Hạn chế của đề tài** 55](#_Toc200398098)

[**3.3. Định hướng mở rộng** 56](#_Toc200398099)

[**Tài liệu tham khảo** 59](#_Toc200398100)

[Hình 1.4.2. Thư mục chứa file ảnh hoa 11](#_Toc200398625)

[Hình 1.4.4. Biểu đồ phân bố số lượng ảnh theo lớp hoa trong tập huấn luyện 13](#_Toc200398626)

[Hình 2.1.1. Hình mảng ma trận RGB 6x6x3 15](#_Toc200398627)

[Hình 2.1.3. Mô tả bước đầu tiên của quá trình tích chập 17](#_Toc200398628)

[Hình 2.1.4. Mô tả tính chất hàm ReLu 18](#_Toc200398629)

[Hình 2.1.5. Mô tả Max Pooling 19](#_Toc200398630)

[Hình 2.1.6. Khối Residual cơ bản 22](#_Toc200398631)

[Hình 2.2.1. Các thư viện cần thiết 24](#_Toc200398632)

[Hình 2.2.2. Dữ liệu đầu vào 25](#_Toc200398633)

[Hình 2.2.3. Mô hình Simple CNN 25](#_Toc200398634)

[Hình 2.2.4. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa 27](#_Toc200398635)

[Hình 2.2.5. Huấn luyện mô hình CNN 28](#_Toc200398636)

[Hình 2.2.6. Đánh giá mô hình CNN 29](#_Toc200398637)

[Hình 2.2.6. Hình lớp BasicBlock 30](#_Toc200398638)

[Hình 2.2.7. Mô hình ResNet18 31](#_Toc200398639)

[Hình 2.2.8. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa 33](#_Toc200398640)

[Hình 2.2.9. Huấn luyện mô hình ResNet18 34](#_Toc200398641)

[Hình 2.2.10. Đánh giá mô hình ResNet18 35](#_Toc200398642)

[Hình 2.3.1. Hình thể hiện chỉ số Loss, Accuracy của từng epoch CNN 36](#_Toc200398643)

[Hình 2.3.2. Hình thể hiện các chỉ số đánh giá mô hình CNN 38](#_Toc200398644)

[Hình 2.3.4. Biểu đồ thể hiện đồ thị mất mát và độ chính xác CNN 40](#_Toc200398645)

[Hình 2.3.5. Ma trận nhầm lẫn CNN 42](#_Toc200398646)

[Hình 2.3.6. Hình thể hiện chỉ số Loss, Accuracy của từng epoch ResNet18 44](#_Toc200398647)

[Hình 2.3.7. Hình thể hiện các chỉ số đánh giá mô hình ResNet18 46](#_Toc200398648)

[Hình 2.3.8. Biểu đồ thể hiện đồ thị mất mát và độ chính xác ResNet18 48](#_Toc200398649)

[Hình 2.3.9. Ma trận nhầm lẫn ResNet18 50](#_Toc200398650)

[Bảng 2.1. Mô tả kiến trúc chi tiết CNN 21](#_Toc200401646)

[Bảng 2.2. Mô tả kiến trúc mô hình ResNet18 24](#_Toc200401647)

**CHƯƠNG 1. GIỚI THIỆU BÀI TOÁN**

**1.1. Mô tả bài toán**

Trong kỷ nguyên phát triển nhanh chóng của khoa học công nghệ, đặc biệt là Trí tuệ nhân tạo (AI) và Học máy (Machine Learning), khả năng của máy tính trong việc tự học và đưa ra quyết định đang tạo ra nhiều đột phá trong các lĩnh vực thực tiễn. Một trong những lĩnh vực nổi bật nhất chính là **Thị giác máy tính (Computer Vision)** – nơi máy tính được huấn luyện để “nhìn thấy” và “hiểu” hình ảnh, tương tự như cách con người cảm nhận thế giới thị giác.

1.1.1. Tổng quan bài toán phân loại hình ảnh

Phân loại hình ảnh (Image Classification) là một trong những bài toán nền tảng và quan trọng nhất trong lĩnh vực thị giác máy tính (Computer Vision). Bài toán đặt ra yêu cầu xây dựng một mô hình đầu vào là một ảnh số và đầu ra là nhãn phân loại đại diện cho đối tượng chính trong ảnh. Các mô hình học sâu hiện đại có thể tự động học đặc trưng từ dữ liệu ảnh đầu vào, giúp cải thiện đáng kể độ chính xác so với các phương pháp truyền thống dựa trên đặc trưng thủ công (hand-crafted features).

1.1.2. Bài toán phân loại hoa

**Phân loại hình ảnh hoa (Flower Image Classification)** – một bài toán cơ bản nhưng đầy thách thức trong thị giác máy tính. Cụ thể, mục tiêu là xây dựng mô hình có khả năng phân biệt **102 loài hoa khác nhau** dựa trên hình ảnh đầu vào, từ đó gán nhãn đúng cho từng ảnh hoa.

Đây là một bài toán mang tính chất **phân loại cấp độ chi tiết (fine-grained classification)**, nơi mà sự khác biệt giữa các lớp (loài hoa) có thể rất tinh vi, đòi hỏi mô hình học sâu phải trích xuất và học được các đặc trưng hình học, màu sắc và cấu trúc cụ thể như hình dạng cánh hoa, độ đậm nhạt của màu sắc, hình thái nhụy và nhiều cấu trúc khác.

1.1.3. Mục tiêu nghiên cứu

Nghiên cứu và áp dụng các mô hình học sâu hiện đại để giải quyết bài toán cụ thể là mạng Nơ-ron tích chập (CNN) và mô hình kỹ thuật học chuyển giao dựa trên ResNet18 để giải quyết bài toán phân loại hình ảnh hoa.

Xây dựng và huấn luyện một mô hình CNN cơ bản (CNN) từ đầu, đồng thời triển khai một mô hình sử dụng kỹ thuật Học chuyển giao (Transfer Learning) dựa trên kiến trúc ResNet18.

Đánh giá hiệu suất của các mô hình đã xây dựng thông qua các chỉ số định lượng như độ chính xác (Accuracy), hàm mất mát (Loss) và ma trận nhầm lẫn (Confusion Matrix).

So sánh và phân tích ưu nhược điểm của các phương pháp tiếp cận, từ đó rút ra nhận xét về hiệu quả của từng mô hình trong bài toán cụ thể này.

1.1.4. Ứng dụng thực tế của bài toán phân loại hoa

Sinh học và thực vật học: Hỗ trợ các nhà khoa học trong việc nhận diện và phân loại nhanh chóng các loài thực vật, đặc biệt là trong các dự án nghiên cứu đa dạng sinh học hoặc bảo tồn loài.

Nông nghiệp và làm vườn: Giúp người nông dân hoặc người làm vườn dễ dàng nhận biết các loại hoa, theo dõi tình trạng cây trồng, hoặc phát hiện sớm các dấu hiệu bệnh tật dựa trên hình thái của hoa.

Giáo dục và giải trí: Phát triển các ứng dụng di động cho phép người dùng chụp ảnh hoa và nhận được thông tin về loài hoa đó, phục vụ mục đích học tập, tìm hiểu tự nhiên hoặc đơn giản là thỏa mãn sự tò mò.

Thương mại và du lịch: Ứng dụng trong việc quản lý sản phẩm hoa, định giá, hoặc phát triển các tour du lịch sinh thái tập trung vào các loài hoa đặc trưng của một vùng.

Chăm sóc cây cảnh: Xác định loại hoa để đưa ra hướng dẫn chăm sóc phù hợp.

**1.2. Bộ dữ liệu hình ảnh hoa**

Bộ dữ liệu tên là Oxford 102 Category Flower Dataset với đường dẫn (https://www.kaggle.com/datasets/waseemalastal/the-oxford-flowers-102-dataset).

Bộ dữ liệu do nhóm nghiên cứu Visual Geometry Group (VGG), Đại học Oxford phát triển và công bố lần đầu vào năm 2008. Đây là một trong những bộ dữ liệu chuẩn trong các nghiên cứu học máy về phân loại ảnh thực vật. Bộ dữ liệu đã được tải về cục bộ và tổ chức theo cấu trúc thư mục phù hợp cho việc nạp dữ liệu bằng PyTorch.

Cách gán nhãn và cân bằng dữ liệu:

* Các hình ảnh trong bộ dữ liệu được gán nhãn theo cấu trúc thư mục. Mỗi thư mục con trong tập huấn luyện (train) và tập xác thực (valid) tương ứng với một lớp hoa cụ thể. Tên của thư mục con đó được sử dụng làm nhãn lớp cho tất cả các hình ảnh bên trong nó. Ví dụ, tất cả các hình ảnh trong thư mục 001 sẽ được gán nhãn là lớp hoa thứ nhất (ví dụ: pink primrose), thư mục 002 là lớp hoa thứ hai (ví dụ: hard-leaved pocket orchid), và cứ thế tiếp tục cho đến 102 lớp.
* Về mặt cân bằng dữ liệu, bộ dữ liệu Oxford 102 Flowers được thiết kế tương đối cân bằng về số lượng mẫu giữa 102 lớp khác nhau, với khoảng 40 đến 258 hình ảnh cho mỗi lớp. Tuy nhiên, vẫn có sự chênh lệch nhỏ về số lượng mẫu giữa các lớp. Việc trực quan hóa phân bố nhãn ở mục 1.4 sẽ cung cấp cái nhìn cụ thể hơn về sự cân bằng này. Dữ liệu cân bằng giúp mô hình không bị thiên vị về các lớp có nhiều mẫu hơn.

**1.3. Mô tả dữ liệu**

Bộ dữ liệu hoa được tổ chức thành hai tập chính: tập huấn luyện (training set) và tập xác thực (validation set), được lưu trữ trong các thư mục riêng biệt.

* Định dạng hình ảnh: Tất cả các hình ảnh trong bộ dữ liệu đều ở định dạng JPEG (.jpg).
* Số kênh màu: Các hình ảnh là ảnh màu (RGB - Red, Green, Blue), do đó mỗi hình ảnh có 3 kênh màu.
* Kích thước hình ảnh gốc: Kích thước (chiều rộng x chiều cao) của các hình ảnh trong bộ dữ liệu gốc là không đồng nhất. Để phù hợp với yêu cầu đầu vào của các mô hình học sâu và chuẩn hóa dữ liệu, tất cả các hình ảnh sẽ được tiền xử lý bằng cách thay đổi kích thước về một kích thước cố định (128x128 pixels cho SimpleCNN và 224x224 pixels cho ResNet18).
* Số lượng lớp: Bộ dữ liệu bao gồm tổng cộng 102 lớp (loài) hoa khác nhau.
* Số lượng mẫu dữ liệu:
  + Tập huấn luyện (Training Set): Bao gồm 6552 hình ảnh. Đây là tập dữ liệu mà mô hình sẽ học từ đó để nhận diện các đặc trưng của từng lớp hoa.
  + Tập xác thực (Validation Set): Bao gồm 818 hình ảnh. Tập này được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện, giúp theo dõi sự hội tụ và phát hiện hiện tượng quá khớp (overfitting). Mặc dù có số lượng ảnh lớn hơn tập huấn luyện, đây là một cách chia dữ liệu phổ biến trong các bộ dữ liệu thách thức như Oxford 102 Flowers để đảm bảo đánh giá khách quan.
  + Tập kiểm tra (Test): Bao gồm 819 hình ảnh, được sử dụng nhằm đánh giá khách quan hiệu suất của mô hình sau khi đã hoàn thành quá trình huấn luyện và điều chỉnh trên tập huấn luyện (train) và tập xác thực (validation). Các hình ảnh trong tập kiểm tra là những dữ liệu mà mô hình chưa từng nhìn thấy trong quá trình học, giúp phản ánh chính xác khả năng tổng quát hóa của mô hình đối với dữ liệu mới. Kết quả trên tập kiểm tra thường được dùng để báo cáo độ chính xác cuối cùng của mô hình.

**1.4. Trực quan dữ liệu**

A collage of different flowers

AI-generated content may be incorrect.Hình 1.4.1. Một số ảnh mẫu từ bộ dữ liệu hoa

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 1.4.2. Thư mục chứa file ảnh hoa

A screenshot of a computer

AI-generated content may be incorrect.Hình 1.4.3. Cấu trúc thư mục phân lớp file ảnh hoa

A graph of blue lines

AI-generated content may be incorrect.

Hình 1.4.4. Biểu đồ phân bố số lượng ảnh theo lớp hoa trong tập huấn luyện

**CHƯƠNG 2. XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH**

**2.1. Mô tả kiến trúc của các mô hình học sâu**

Ở đề tài này, hai mô hình học sâu được lựa chọn để thực hiện bài toán phân loại hoa:

* Mô hình mạng Nơ-ron tích chập đơn giản (SimpleCNN) tự xây dựng.
* Mô hình ResNet18 sử dụng kỹ thuật học chuyển giao (Transfer Leaning).

2.1.1. Mô hình SimpleCNN

2.1.1.1. Tổng quan về mô hình CNN

Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

A group of squares with a black text

AI-generated content may be incorrect.

Hình 2.1.1. Hình mảng ma trận RGB 6x6x3

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

A diagram of a diagram of a diagram

AI-generated content may be incorrect.Hình 2.1.2. Kiến trúc tổng quan của một mạng Nơ-ron tích chập

2.1.1.2. Kiến trúc của mô hình CNN

CNN bao gồm tập hợp các lớp cơ bản bao gồm: Convolution layer + Nonlinear layer (Relu), Pooling layer, Fully connected layer. Các lớp này liên kết với nhau theo một thứ tự nhất định. Thông thường, một ảnh sẽ được lan truyền qua tầng Convolution layer + Nonlinear layer (Relu) đầu tiên, sau đó các giá trị tính toán được sẽ lan truyền qua pooling layer, bộ ba Convolution layer + Nonlinear layer (Relu) + Pooling layer có thể được lặp lại nhiều lần trong network. Và sau đó được lan truyền qua tầng fully connected layer và softmax để tính sác xuất ảnh đó chứa vật thế gì.

Cấu trúc của CNN như sau: Đầu vào là một ảnh màu kích thước 128x128x3. Ảnh được lan truyền qua các tầng cơ bản sau:

1. Convolution 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.

A screenshot of a computer game

AI-generated content may be incorrect.

Hình 2.1.3. Mô tả bước đầu tiên của quá trình tích chập

1. Nonlinear Layer: là lớp sử dụng một hàm kích hoạt phi tuyến (nonlinear activation function) để biến đổi đầu ra của các lớp trước đó. ReLU là hàm kích hoạt phổ biến nhất cho CNN.Hàm ReLU được ưa chuộng vì tính toán đơn giản, giúp hạn chế tình trạng vanishing gradient, và cũng cho kết quả tốt hơn. ReLU cũng như những hàm kích hoạt khác, được đặt ngay sau tầng convolution, ReLU sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0.

ReLU cũng có một số vấn đề tiềm ẩn như không có đạo hàm tại điểm 0, giá trị của hàm ReLU có thể lớn đến vô cùng và nếu chúng ta không khởi tạo trọng số cẩn thận, hoặc khởi tạo learning rate quá lớn thì những neuron ở tầng này sẽ rơi vào trạng thái chết, tức là luôn có giá trị < 0.

A cartoon of a person with arms extended

AI-generated content may be incorrect.

Hình 2.1.4. Mô tả tính chất hàm ReLu

1. 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.

A diagram of a graph

AI-generated content may be incorrect.

Hình 2.1.5. Mô tả Max Pooling

1. 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.)

2.1.1.3. Kiến trúc chi tiết

Đối với các mô hình cơ bản số lớp tích chập thường sao động từ 2 đến 5 lớp. Số lớp fully connected (kết nối đầy đủ) thường nằm trong khoảng từ 1 đến 3 lớp. Kích thước ảnh đầu vào cũng rất đa dạng, phụ thuộc vào yêu cầu của bài toán và bộ dữ liệu, phổ biến nhất là các kích thước 28x28, 32x32, 64x64, 128x128 hoặc 224x224 pixel.

Với bài toán thì tổng lớp tích chập 3, lớp Pooling 3, lớp Fully connected 2:

|  |  |  |
| --- | --- | --- |
| Tầng | Cấu trúc đầu ra | Ghi chú |
| |  | | --- | | Conv2d(3, 32, p=1) |  |  | | --- | |  | | 128x128x32 | Tích chập |
| ReLu + MaxPool(2) | 64x64x32 | Giảm một nửa kích thước không gian |
| Conv2d(32, 64, 3, p=1) | 64x64x64 | Tăng số lượng đặc trưng |
| ReLu + MaxPool(2) | 32x32x3=64 |  |
| Con2d(64, 128, 3, p=1) | 32x32x128 |  |
| ReLU + MaxPool(2) | 16x16x128 |  |
| Flatten | 32768 | 128x16x16=32768 |
| Linear(32768, 512) | 512 | Fully connected |
| Dropout(0.5) | 512 | Giảm overfiting |
| Linear(512, 102) | 102 | Đầu ra ứng với 102 loài hoa |

Bảng 2.1. Mô tả kiến trúc chi tiết CNN

2.1.2. Mô hình ResNet18

Mô hình resnet được giới thiệu bởi Kaiming He và các cộng sự trong bài báo nổi tiếng “Deep Residual Learning for Image Recognition” vào năm 2015. Ý tưởng đột phá của ResNet là sử dụng skip connections hoặc residual connections giữa các lớp để giải quyết vấn đề suy giảm độ chính xác khi mô hình trở nên sâu hơn. Điều này cho phép thông tin được truyền qua các lớp của mạng mà không bị suy giảm, giúp mô hình có thể học được các đặc trưng (features) phức tạp từ dữ liệu.

2.1.2.1. Kiến trúc mô hình ResNet18

Kiến trúc ResNet-18 là một ví dụ tiêu biểu về mạng nơ-ron tích chập (CNN) hiện đại, ứng dụng các lớp cơ bản như Convolution layer, Nonlinear layer (ReLU), Pooling layer, Fully connected layer và đặc biệt là các khối Residual giúp khắc phục hiện tượng mất mát thông tin khi mạng sâu. Các lớp này liên kết với nhau theo một trình tự nhất định, tạo nên một mạng lưới mạnh mẽ và hiệu quả cho các bài toán nhận diện ảnh.

Cụ thể, đầu vào của ResNet-18 là một ảnh màu với kích thước phổ biến như 224x224x3 (3 kênh RGB). Ảnh này sẽ lần lượt đi qua các tầng cơ bản sau:

1. ConvolutionLayer: Lớp tích chập đầu tiên sử dụng kernel kích thước lớn (7x7) và stride 2 để trích xuất đặc trưng tổng quát, sau đó là một lớp Batch Normalization và hàm kích hoạt ReLU. Kết quả tiếp tục qua lớp Max Pooling để giảm kích thước feature map, giúp tăng tốc độ tính toán và giảm số tham số.
2. Các khối Residual: ResNet-18 gồm 4 nhóm (stage) residual block, mỗi nhóm có 2 block, mỗi block gồm 2 lớp tích chập 3x3, xen kẽ là các lớp Batch Normalization và hàm kích hoạt ReLU. Điểm nổi bật là các khối residual có thêm kết nối tắt (skip connection), cho phép tín hiệu đầu vào đi thẳng qua các block mà không bị suy giảm, giúp mạng sâu hơn nhưng vẫn dễ huấn luyện.

A diagram of a diagram

AI-generated content may be incorrect.

Hình 2.1.6. Khối Residual cơ bản

1. Nonlinear Layer (ReLU): Sau mỗi phép tích chập, hàm kích hoạt ReLU được áp dụng để loại bỏ giá trị âm, tăng tính phi tuyến và giúp mô hình học hiệu quả hơn.
2. Pooling Layer: Ngoài lớp Max Pooling đầu tiên, cuối mạng còn sử dụng Average Pooling toàn cục (Global Average Pooling) để tổng hợp thông tin từ toàn bộ feature map thành một vector duy nhất, giảm tối đa số tham số trước khi đưa vào lớp phân loại.
3. Fully Connected Layer:Sau khi đã làm phẳng (flatten) kết quả từ các lớp trước, ResNet-18 sử dụng một lớp fully connected (dense) để thực hiện nhiệm vụ phân loại cuối cùng. Đầu ra của lớp này là số lượng lớp cần phân loại (ví dụ: 1000 lớp với ImageNet), sử dụng hàm kích hoạt Softmax để tính toán xác suất cho từng lớp đầu ra.

2.1.2.2. Các lớp trong ResNet-18 bao gồm:

|  |  |  |  |
| --- | --- | --- | --- |
| Tầng | Cấu hình lớp | Đầu ra | Ghi chú |
| Input |  | 224x224x3 | |  | | --- | |  |   Ảnh đầu vào là ảnh chuẩn RGB Ảnh đầu vào là ảnh chuẩn RGB |
| Conv1 | Conv2d(3, 64, kernel\_size=7, stride=2, padding=3) | 112×112×64 | Tích chập đầu tiên, giảm kích thước |
| BatchNorm2d + ReLU | BatchNorm2d + ReLU | 112×112×64 | Chuẩn hóa và kích hoạt |
| MaxPool | MaxPool2d(kernel\_size=3, stride=2, padding=1) | 56×56×64 | Giảm kích thước không gian |
| Conv2\_x | 2 Residual Block: Conv2d(64, 64, 3×3) × 2 | 56x56x64 | Không thay đổi kích thước stride=1 |
| Conv3\_x | 2 Residual Block: Conv2d(64, 128, 3×3, stride=2) → Conv2d(128, 128, 3×3) | 28x28x128 | Tăng depth + giảm resolution |
| Conv4\_x | 2 Residual Block: Conv2d(128, 256, 3×3, stride=2) → Conv2d(256, 256, 3×3) | 14x14x256 |  |
| Conv5\_x | 2 Residual Block: Conv2d(256, 512, 3×3, stride=2) → Conv2d(512, 512, 3×3) | 7x7x512 |  |
| AvgPool | AdaptiveAvgPool2d((1,1)) | 1x1x512 | Tổng hợp đặc trưng toàn cục |
| FC (Linear) | Linear(in\_features=512, out\_features=102) | 102 | Lớp đầu ra cho phân loại 102 loài hoa |

Bảng 2.2. Mô tả kiến trúc mô hình ResNet18

**2.2. Cài đặt mô hình bằng Python**

Các thư viện cần thiết để cài đặt 2 mô hình:

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 2.2.1. Các thư viện cần thiết

Chuẩn bị dữ liệu:

A close up of a computer screen

AI-generated content may be incorrect.

Hình 2.2.2. Dữ liệu đầu vào

2.2.1. Cài đặt mô hình SimpleCNN

2.2.1.1. Lớp SimpleCNN

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 2.2.3. Mô hình Simple CNN

**Giải thích code:**

1. Class SimpleCNN(nn.Module): khởi tạo lớp SimpleCNN kế thừa từ lớp nn.Modul lớp cơ bản của các mô hình PyTorch.
2. def \_\_init\_\_(self, num\_classes=102): hàm khởi tạo với số lớp đầu ra là 102 .
3. seft.features = nn.Sequential(): các lớp tích chập.

nn.Conv2d(3, 32, kernel\_size=3, padding=1): Lớp tích chập đầu tiên.

nn.Conv2d(32, 64, kernel\_size=3, padding=1): Lớp tích chập thứ hai: 32 -> 64 bộ lọc.

nn.Conv2d(64, 128, kernel\_size=3, padding=1): Lớp tích chập thứ ba: 64 -> 128 bộ lọc.

nn.ReLU(): Hàm kích hoạt rectified linear unit, chuyển đổi các giá trị âm thành 0.

nn.MaxPool2d(2): Lớp gộp cực đại với kích thước kernel 2, giảm kích thước ảnh xuống một nửa.

1. self.classifier = nn.Sequential(): phân loại.

nn.Flatten(): Làm phẳng đầu ra từ các lớp tích chập thành vector 1D.

nn.Linear(128 \* 16 \* 16, 512): Lớp kết nối đầy đủ (Dense/Linear) đầu tiên: 128\*16\*16 đầu vào, 512 đầu ra.

nn.ReLU(): Hàm kích hoạt rectified linear unit, chuyển đổi các giá trị âm thành 0.

nn.Dropout(0.5): Kỹ thuật Dropout để chống quá khớp, loại bỏ ngẫu nhiên 50% nơ-ron.

nn.Linear(512, num\_classes): Lớp kết nối đầy đủ cuối cùng: 512 đầu vào, num\_classes (số lớp hoa) đầu ra.

1. def forward(self, x): Luồng truyền dữ liệu qua mạng.

2.2.1.2. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa

A computer code with black text

AI-generated content may be incorrect.

Hình 2.2.4. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa

* device: Xác định thiết bị tính toán (GPU hoặc CPU).
* model\_cnn: Khởi tạo đối tượng SimpleCNN.
* criterion\_cnn: Định nghĩa hàm mất mát (CrossEntropyLoss).
* optimizer\_cnn: Định nghĩa thuật toán tối ưu hóa (Adam).

2.2.1.3. Huấn luyện và đánh giá mô hình CNN

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 2.2.5. Huấn luyện mô hình CNN

* Vòng lặp chính huấn luyện mô hình qua các epoch.
* Chế độ huấn luyện (model.train()): Bật các lớp như Dropout và BatchNorm để hoạt động bình thường.
* optimizer.zero\_grad(): Xóa các gradient của các tham số.
* model(inputs): Thực hiện forward pass.
* loss.backward(): Tính toán gradient của hàm mất mát.
* optimizer.step(): Cập nhật trọng số của mô hình.
* Chế độ đánh giá (model.eval()): Tắt Dropout và BatchNorm hoạt động ở chế độ inference.
* torch.no\_grad(): Vô hiệu hóa tính toán gradient để tiết kiệm bộ nhớ và tăng tốc độ.
* Tính toán độ chính xác trên tập validation.

A computer code with many letters

AI-generated content may be incorrect.

Hình 2.2.6. Đánh giá mô hình CNN

2.2.2. Cài đặt mô hình ResNet18

Nguồn tham khảo:

<https://www.geeksforgeeks.org/resnet18-from-scratch-using-pytorch/>

2.2.2.1. Lớp BasicBlock

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 2.2.6. Hình lớp BasicBlock

**Giải thích code:** Cấu trúc cốt lõi của ResNet, với phép tích chập, chuẩn hóa theo lô, kích hoạt ReLU.

* class BasicBlock(nn.Module): Kế thừa từ nn.Module, là khối cơ bản trong ResNet18.
* expansion = 1: Mặc định là 1 cho BasicBlock.
* def \_\_init\_\_(self, in\_channels, out\_channels, stride=1, downsample=None): hàm khởi tạo.
* conv1: tích chập đầu tiên, có thể giảm kích thước.
* bn1, bn2: chuẩn hóa giúp tăng ổn định trong huấn luyện.
* relu: hàm kích hoạt phi tuyến.
* conv2: tích chập thứ hai, stride luôn = 1.
* downsample: dùng để chuyển đổi identity nếu kích thước đầu vào khác đầu ra.
* def forward(self, x): Luồng truyền dữ liệu qua mạng.

2.2.2.2. Mô hình ResNet18

A screenshot of a computer program

AI-generated content may be incorrect.A white background with black dots

AI-generated content may be incorrect.

Hình 2.2.7. Mô hình ResNet18

**Giải thích code:**

Phần khởi tạo: def\_\_innit\_\_

* Khai báo lớp kế thừa từ nn.Module.
* num\_classes: số lớp phân loại đầu ra (mặc định là 102).
* self.in\_channels = 64: số kênh đầu ra của conv đầu tiên (giữ để chuyển cho các block tiếp theo).
* self.conv1: lớp tích chập đầu vào, nhận ảnh RGB (3 kênh), kernel 7x7, stride 2, padding 3.
* self.bn1: batch normalization cho 64 kênh đầu ra từ conv1.
* self.relu: hàm kích hoạt ReLU.
* self.maxpool: lớp pooling 3x3, stride 2, giảm kích thước không gian.
* self.layer1 → layer4: 4 nhóm residual block với số bộ lọc tăng dần (64, 128, 256, 512), mỗi nhóm có 2 block.
* stride=2 ở layer2, layer3, layer4 giúp giảm kích thước feature map.
* self.avgpool: global average pooling, đưa feature map còn lại về kích thước 1x1.
* self.fc: fully connected layer, nhận input 512 chiều (số kênh cuối cùng) và xuất ra số lớp phân loại.

Hàm tạo Layer (\_make\_layer):

* downsample: nếu stride ≠ 1 hoặc số kênh không khớp, cần một conv 1x1 để biến đổi kích thước cho nhánh skip connection.
* layers: danh sách các block, block đầu có thể thay đổi kích thước, các block sau giữ nguyên.
* self.in\_channels = out\_channels: cập nhật số kênh cho các block tiếp theo.
* return nn.Sequential(\*layers): trả về một module gồm nhiều block liên tiếp.

Hàm lan truyền tiến (forward):

* x = self.conv1(x) → self.maxpool(x): trích xuất đặc trưng ban đầu và giảm kích thước.
* x = self.layer1(x) → self.layer4(x): lần lượt truyền qua các residual block.
* x = self.avgpool(x): pooling toàn cục.
* x = torch.flatten(x, 1): làm phẳng tensor về dạng (batch\_size, 512).
* x = self.fc(x): lớp fully connected để phân loại.
* return x: trả về đầu ra mô hình (logits cho từng lớp).

2.2.2.3. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa

A screen shot of a computer code

AI-generated content may be incorrect.

Hình 2.2.8. Khởi tạo mô hình, hàm mất mát, bộ tối ưu hóa

2.2.2.4. Huấn luyện và đánh giá mô hình ResNet18

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 2.2.9. Huấn luyện mô hình ResNet18

* num\_epochs\_resnet = 50: Số lần lặp huấn luyện là 50.
* train\_losses\_resnet = [] và val\_accuracies\_resnet = []: Khởi tạo hai danh sách rỗng để lưu trữ giá trị hàm mất mát (loss) trên tập huấn luyện và độ chính xác (accuracy) trên tập validation sau mỗi epoch. Điều này hữu ích cho việc vẽ đồ thị.
* for epoch in range(num\_epochs\_resnet): vòng lặp chính của huấn luyện
* Chế độ huấn luyện (model.train()): Bật các lớp như Dropout và BatchNorm để hoạt động bình thường.
* optimizer.zero\_grad(): Xóa các gradient của các tham số.
* model(inputs): Thực hiện forward pass.
* loss.backward(): Tính toán gradient của hàm mất mát.
* optimizer.step(): Cập nhật trọng số của mô hình.
* Chế độ đánh giá (model.eval()): Tắt Dropout và BatchNorm hoạt động ở chế độ inference.
* torch.no\_grad(): Vô hiệu hóa tính toán gradient để tiết kiệm bộ nhớ và tăng tốc độ.
* Tính toán độ chính xác trên tập validation.

A screenshot of a computer code

AI-generated content may be incorrect.

Hình 2.2.10. Đánh giá mô hình ResNet18

**2.3. Đánh giá mô hình CNN và ResNet18**

2.3.1. Đánh giá mô hình CNN

2.3.1.1. Tỉ lệ chính xác Accuracy

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.3.1. Hình thể hiện chỉ số Loss, Accuracy của từng epoch CNN

A table of numbers with numbers

AI-generated content may be incorrect.

A table of numbers with numbers in the middle

AI-generated content may be incorrect.

Hình 2.3.2. Hình thể hiện các chỉ số đánh giá mô hình CNN

Ở hình 2.3.1 và 2.3.2 đã thể hiện Accuracy giao động nhiều nhất ở khoảng 0.45 – 0.50.

* + Tổng thể Accuracy là 0.45
  + Macro Avg Precision: 0.42
  + Macro Avg Recall: 0.40
  + Macro Avg F1-score: 0.38
  + Weighted Avg Precision: 0.47
  + Weighted Avg Recall: 0.45
  + Weighted Avg F1-score: 0.44
  + Tổng số mẫu (support): 818

Từ các chỉ số ta có thể thấy rằng:

* Độ chính xác tổng thể (accuracy) của mô hình SimpleCNN trên tập validation là khoảng 45.11%. Đây là một con số tương đối thấp, đặc biệt khi có tới 102 lớp hoa. Một độ chính xác thấp như vậy cho thấy mô hình gặp nhiều khó khăn trong việc phân biệt các loại hoa.
* Các chỉ số precision, recall, f1-score trung bình (cả macro và weighted) cũng đều dưới 0.5 (hoặc 50%), khẳng định hiệu suất tổng thể của mô hình chưa tốt.
* Khi xem xét chi tiết từng lớp trong báo cáo phân loại (Hình 2.3.2), có rất nhiều lớp có precision, recall, và f1-score bằng 0.00. Điều này có nghĩa là mô hình không dự đoán đúng bất kỳ mẫu nào cho các lớp đó hoặc thậm chí không bao giờ dự đoán các mẫu thuộc về các lớp đó. Ví dụ, lớp 0, 3, 10, 13, 21, 24, 28, 35, 69, 79, 86, 91, 95... đều có chỉ số là 0.00. Điều này chỉ ra rằng mô hình không học được cách phân biệt các lớp này.
* Ngược lại, có một số lớp có chỉ số khá tốt (ví dụ: lớp 1, 8, 14 có precision, recall, f1-score là 1.00 hoặc gần 1.00), nhưng những trường hợp này thường có support (số lượng mẫu) rất thấp (ví dụ: lớp 1 có support là 1, lớp 8 có support là 1, lớp 14 có support là 6). Điều này có thể là do sự trùng hợp ngẫu nhiên hoặc các lớp này quá dễ phân biệt, chứ không phải do mô hình học được một cách tổng quát.

2.3.1.2. Nhận xét hàm mất mát và độ chính xác CNN

A graph of a graph of a graph

AI-generated content may be incorrect.

Hình 2.3.4. Biểu đồ thể hiện đồ thị mất mát và độ chính xác CNN

Từ hình 2.3.4

Với hàm mất mát huấn luyện (Train Loss):

* Loss huấn luyện giảm rất nhanh trong khoảng 10-15 epoch đầu tiên, từ giá trị cao ban đầu (khoảng 4.0) xuống dưới 0.5.
* Sau đó, loss tiếp tục giảm chậm và ổn định ở mức rất thấp, gần như bằng 0 (khoảng 0.1) sau khoảng 30 epoch.
* Sự giảm mạnh và duy trì ở mức thấp của Train Loss cho thấy mô hình đang học rất tốt trên tập dữ liệu huấn luyện, có thể đang học thuộc lòng dữ liệu.

Với độ chính xác trên tập Validation (Validation Accuracy):

* Accuracy validation tăng khá nhanh trong khoảng 10-15 epoch đầu, từ khoảng 0.15 lên khoảng 0.45-0.50.
* Tuy nhiên, sau khoảng epoch 20, độ chính xác validation chững lại và dao động trong khoảng 0.45 đến 0.50 (hoặc 45% - 50%), không có sự cải thiện đáng kể nào nữa.
* Trong khi đó, Train Loss vẫn tiếp tục giảm xuống rất thấp.

**Kết luận:**

Đồ thị này là dấu hiệu rõ ràng của Overfitting (Quá khớp):

* Train Loss giảm rất thấp cho thấy mô hình đang học thuộc lòng dữ liệu huấn luyện một cách hiệu quả.
* Validation Accuracy chững lại ở mức thấp và dao động cho thấy mô hình không thể khái quát hóa những gì đã học sang dữ liệu mới (tập validation).
* Sự chênh lệch lớn giữa hiệu suất trên tập huấn luyện (loss rất thấp) và tập validation (accuracy thấp) là minh chứng điển hình của overfitting.

2.3.1.3. Ma trận nhầm lẫn

A graph showing a line

AI-generated content may be incorrect.

Hình 2.3.5. Ma trận nhầm lẫn CNN

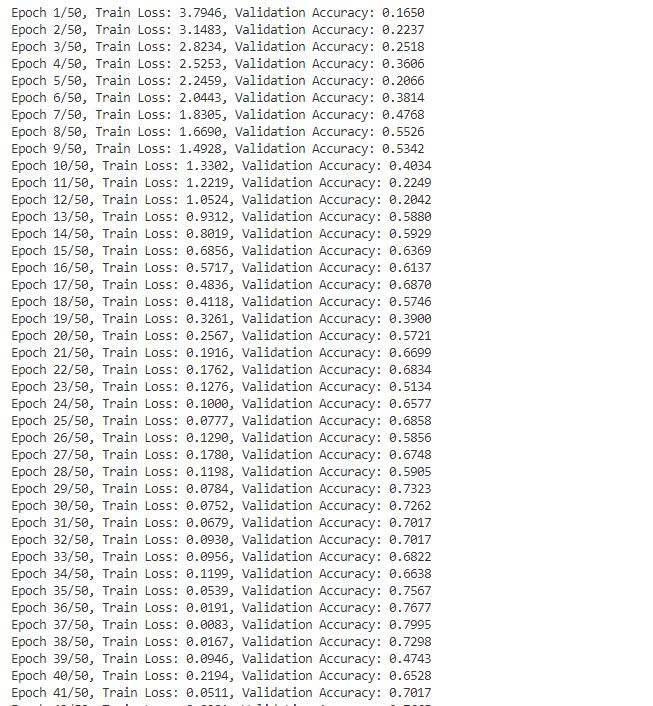
**Mô tả:** Đây là một ma trận 102x102 (tương ứng với 102 lớp hoa). Các ô trên đường chéo chính (từ trên cùng bên trái xuống dưới cùng bên phải) đại diện cho các dự đoán đúng. Các ô ngoài đường chéo đại diện cho các dự đoán sai.

**Nhận xét:**

* Đường chéo chính có màu xanh đậm hơn, cho thấy có một số dự đoán đúng. Tuy nhiên, màu sắc trên đường chéo không đồng đều và không đủ đậm trên toàn bộ ma trận, cho thấy nhiều lớp không được phân loại đúng.
* Có rất nhiều ô ngoài đường chéo có màu xanh nhạt hoặc trắng, chứng tỏ số lượng dự đoán sai rất lớn.
* Đáng chú ý, có nhiều hàng và cột hoàn toàn trắng (không có bất kỳ điểm xanh nào), đặc biệt là các hàng và cột ở gần cuối. Điều này phản ánh lại kết quả trong báo cáo phân loại, nơi nhiều lớp có precision, recall, f1-score bằng 0. Tức là, mô hình không thể phân loại đúng hoặc thậm chí không dự đoán bất kỳ mẫu nào thuộc về những lớp đó.
* Sự phân bố các điểm xanh trên ma trận cho thấy mô hình đang gặp khó khăn trong việc phân biệt giữa các lớp hoa khác nhau.

2.3.2. Đánh giá mô hình ResNet18

2.3.2.1. Tỷ lệ chính xác (Accuracy)



Hình 2.3.6. Hình thể hiện chỉ số Loss, Accuracy của từng epoch ResNet18

A table of numbers with numbers

AI-generated content may be incorrect.A table of numbers with text

AI-generated content may be incorrect.

Hình 2.3.7. Hình thể hiện các chỉ số đánh giá mô hình ResNet18

* Tổng thể Accuracy: 0.74
* Macro Avg Precision: 0.73
* Macro Avg Recall: 0.73
* Macro Avg F1-score: 0.70
* Weighted Avg Precision: 0.78
* Weighted Avg Recall: 0.74
* Weighted Avg F1-score: 0.73
* Tổng số mẫu (support): 818

**Nhận xét:**

* Độ chính xác tổng thể (accuracy) của mô hình ResNet18 trên tập validation là khoảng 73.59%
* Các chỉ số precision, recall, f1-score trung bình (cả macro và weighted) đều cao hơn đáng kể so với SimpleCNN, dao động trong khoảng 0.70 - 0.78
* Khi xem xét chi tiết từng lớp (image\_61a3d5.png và image\_61a3d0.png), mặc dù vẫn còn một số lớp có chỉ số thấp hoặc bằng 0.00 (ví dụ: lớp 40, 41, 46, 50, 52, 53, 56, 59, 63, 64, 65, 68, 70, 72, 79). Hầu hết các lớp đều có precision, recall, f1-score trên 0.50, và nhiều lớp đạt mức rất tốt (ví dụ: lớp 2, 6, 12, 14, 15, 19, 20, 23, 29, 30, 31, 32, 33, 34, 38, 39, 44, 45, 47, 49, 51, 55, 57, 58, 60, 61, 62, 66, 67, 71, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 87, 88, 89, 90, 92, 93, 94, 96, 97, 98, 99, 100, 101 đều có ít nhất một chỉ số trên 0.50)

2.3.2.2. Nhận xét hàm mất mát và độ chính xác ResNet18

A screenshot of a graph

AI-generated content may be incorrect.

Hình 2.3.8. Biểu đồ thể hiện đồ thị mất mát và độ chính xác ResNet18

Hàm mất mát huấn luyện (Train Loss):

* Loss huấn luyện giảm nhanh trong khoảng 15-20 epoch đầu tiên, từ giá trị cao ban đầu (khoảng 3.7) xuống dưới 0.5.
* Sau đó, loss tiếp tục giảm chậm và dao động ở mức rất thấp, gần như bằng 0 (khoảng 0.1) sau khoảng 30 epoch. Có một số đỉnh nhỏ cho thấy sự dao động trong quá trình hội tụ, nhưng xu hướng chung vẫn là giảm.

Độ chính xác trên tập Validation (Validation Accuracy)

* Accuracy validation tăng khá nhanh trong khoảng 15-20 epoch đầu, đạt đến mức khoảng 0.70.
* Sau đó, độ chính xác validation tiếp tục tăng và dao động trong khoảng 0.60 đến 0.80, có những đỉnh cao đạt gần 0.80 (ví dụ: epoch 35, 36, 37, 40, 44, 47).
* So với SimpleCNN, độ chính xác validation của ResNet18 không bị chững lại ở mức thấp mà tiếp tục tăng lên mức cao hơn nhiều.

**Kết luận:**

Đồ thị này cho thấy ResNet18 đã học hiệu quả hơn nhiều và có khả năng khái quát hóa:

* Train Loss giảm đều và Validation Accuracy tăng lên mức cao hơn là dấu hiệu của quá trình học tập thành công.
* Mặc dù có một số dao động lớn trong Validation Accuracy ở các epoch cuối (ví dụ: giảm mạnh ở epoch 39, 43, 49), điều này có thể do kích thước batch nhỏ, sự biến động của dữ liệu trong các batch cuối cùng, hoặc learning rate vẫn còn hơi cao. Tuy nhiên, xu hướng chung vẫn là giữ được độ chính xác ở mức cao.
* So với SimpleCNN, ResNet18 ít bị overfitting nặng nề hơn (mặc dù vẫn có một khoảng cách giữa train loss rất thấp và val accuracy cao nhưng dao động). Điều này cho thấy mô hình học được các đặc trưng khái quát hơn.

2.3.2.3. Ma trận nhầm lẫn

A graph showing a line

AI-generated content may be incorrect.

Hình 2.3.9. Ma trận nhầm lẫn ResNet18

**Mô tả:** Ma trận 102x102 này trực quan hóa hiệu suất phân loại của ResNet18.

**Nhận xét:**

* Đường chéo chính (từ trên cùng bên trái xuống dưới cùng bên phải) có màu xanh đậm hơn và đồng đều hơn nhiều so với ma trận của SimpleCNN. Điều này cho thấy ResNet18 đã phân loại đúng nhiều mẫu và nhiều lớp hơn.
* Các ô ngoài đường chéo chủ yếu là màu xanh rất nhạt hoặc trắng, cho thấy số lượng dự đoán sai đã giảm đi đáng kể. Điều này củng cố rằng mô hình có ít nhầm lẫn hơn giữa các lớp khác nhau.
* Mặc dù vẫn còn một số điểm sáng ngoài đường chéo (cho thấy các lỗi nhầm lẫn), nhưng tổng thể ma trận trông "sạch" hơn nhiều so với SimpleCNN, khẳng định khả năng phân loại tốt hơn của ResNet18.

**2.4. So sánh và giải thích kết quả của mỗi mô hình**

**Hiệu suất tổng thể (Accuracy):**

* SimpleCNN (45.11%) cho thấy hiệu suất kém. Với 102 lớp, 45% dự đoán đúng là không đủ để mô hình có thể ứng dụng thực tế. Nó có thể chỉ tốt hơn một chút so với dự đoán ngẫu nhiên (nếu dự đoán ngẫu nhiên sẽ là khoảng 1/102 ~ 1%).
* Custom ResNet18 (73.59%) là một cải thiện đáng kể. Mặc dù vẫn chưa hoàn hảo, nhưng 73.59% là một con số tương đối tốt cho một bài toán phân loại hình ảnh phức tạp với nhiều lớp như vậy.

**Khả năng phân loại từng lớp:**

* SimpleCNN: Báo cáo chi tiết cho thấy SimpleCNN thất bại hoàn toàn ở rất nhiều lớp (precision/recall/f1-score bằng 0.00). Điều này có nghĩa là mô hình không thể nhận diện được các mẫu thuộc các lớp đó hoặc thậm chí không bao giờ dự đoán bất kỳ mẫu nào thuộc về các lớp đó. Điều này chỉ ra một sự thiếu sót nghiêm trọng trong khả năng học các đặc trưng phân biệt.
* Custom ResNet18: Mặc dù vẫn còn một vài lớp có chỉ số thấp hoặc 0.00, nhưng số lượng các lớp này đã giảm đáng kể. Phần lớn các lớp đều có precision, recall, f1-score trên 0.50, cho thấy ResNet18 có khả năng phân loại hiệu quả hơn trên nhiều loại hoa khác nhau, ngay cả những loại có ít mẫu.

**Đồ thị Learning Curves:**

* SimpleCNN: Biểu đồ Train Loss giảm rất thấp và Validation Accuracy chững lại ở mức thấp (~0.50), là minh chứng rõ ràng của overfitting. Mô hình học thuộc lòng dữ liệu huấn luyện nhưng không thể khái quát hóa cho dữ liệu mới.
* Custom ResNet18: Train Loss cũng giảm rất thấp, nhưng Validation Accuracy tăng lên mức cao hơn nhiều (~0.70-0.80) và duy trì ở đó, mặc dù có một số dao động. Điều này cho thấy ResNet18 có khả năng khái quát hóa tốt hơn và ít bị overfitting nghiêm trọng hơn so với SimpleCNN. Các dao động trong Validation Accuracy ở ResNet18 có thể do dữ liệu batch nhỏ hoặc cần tinh chỉnh thêm learning rate.

**Ma trận nhầm lẫn:**

* SimpleCNN: Ma trận rất "loãng", với nhiều ô ngoài đường chéo đậm màu, cho thấy sự nhầm lẫn lớn giữa các lớp. Đặc biệt, nhiều hàng và cột trắng tinh cho thấy mô hình hoàn toàn bỏ qua hoặc không thể phân loại đúng nhiều lớp.
* Custom ResNet18: Ma trận có đường chéo chính đậm và đồng đều hơn nhiều, với các ô ngoài đường chéo nhạt hơn đáng kể. Điều này minh chứng cho việc mô hình phân loại đúng phần lớn các mẫu và ít bị nhầm lẫn hơn giữa các lớp khác nhau

**Lý do cho sự khác biệt:**

* Kiến trúc: Sự khác biệt lớn nhất nằm ở kiến trúc mạng. SimpleCNN là một mạng nông, không đủ phức tạp để học các đặc trưng tinh tế cần thiết cho việc phân biệt 102 loại hoa có thể rất giống nhau. Nó nhanh chóng đạt giới hạn về khả năng học.
* ResNet18 (với các khối Residual Block và skip connections) được thiết kế để xây dựng các mạng rất sâu mà không gặp phải vấn đề gradient vanishing hay degradation. Các skip connections cho phép thông tin và gradient truyền trực tiếp qua các lớp, giúp mạng học được các biểu diễn đặc trưng phức tạp, đa cấp độ. Batch Normalization cũng đóng vai trò quan trọng trong việc ổn định quá trình huấn luyện và cải thiện khả năng khái quát hóa.

Custom ResNet18 là mô hình vượt trội hơn hẳn so với SimpleCNN trong bài toán phân loại 102 loại hoa. Nó đã thể hiện khả năng học sâu hơn, trích xuất đặc trưng hiệu quả hơn và quan trọng nhất là khả năng khái quát hóa tốt hơn cho dữ liệu chưa từng thấy, điều mà SimpleCNN đã thất bại hoàn toàn do vấn đề overfitting.

**CHƯƠNG 3. KẾT LUẬN**

**3.1. Tóm tắt kết quả đạt được**

Đề tài đã tập trung vào việc xây dựng và đánh giá các mô hình Học Sâu cho bài toán phân loại 102 loại hoa, một nhiệm vụ đầy thách thức do số lượng lớp lớn và sự tương đồng về mặt hình ảnh giữa một số loài hoa. Thông qua quá trình thực hiện, chúng em đã đạt được những kết quả sau:

* Xây dựng và Huấn luyện thành công hai kiến trúc mạng nơ-ron tích chập:
  + Mô hình SimpleCNN: Một kiến trúc mạng tích chập cơ bản đã được thiết kế và huấn luyện.
  + Mô hình Custom ResNet18: Một phiên bản được xây dựng tùy chỉnh dựa trên kiến trúc ResNet18, tận dụng các khối residual và skip connections để học sâu hơn.
* Đánh giá chi tiết hiệu suất của từng mô hình: Chúng tôi đã sử dụng các độ đo tiêu chuẩn như độ chính xác (accuracy), precision, recall, F1-score và ma trận nhầm lẫn để đánh giá hiệu suất trên tập validation. Các đồ thị đường cong học tập (Training Loss và Validation Accuracy) cũng được sử dụng để theo dõi quá trình huấn luyện và chẩn đoán vấn đề.
* Phân tích và So sánh hiệu quả của các kiến trúc:
  + Mô hình SimpleCNN cho thấy hiệu suất còn hạn chế với độ chính xác tổng thể khoảng 45.11% trên tập validation. Đồ thị học tập và ma trận nhầm lẫn chỉ ra rõ ràng vấn đề quá khớp (overfitting) nghiêm trọng, nơi mô hình học thuộc lòng dữ liệu huấn luyện (train loss rất thấp) nhưng không thể khái quát hóa cho dữ liệu mới. Rất nhiều lớp hoa không được phân loại đúng hoặc không được mô hình nhận diện.
  + Mô hình Custom ResNet18 đã chứng minh hiệu suất vượt trội đáng kể với độ chính xác tổng thể khoảng 73.59% trên tập validation. Mặc dù vẫn có một số dao động trong quá trình hội tụ, nhưng mô hình này đã khắc phục được phần lớn vấn đề quá khớp, học được các đặc trưng phức tạp hơn và có khả năng phân loại hiệu quả hơn trên nhiều lớp hoa. Ma trận nhầm lẫn của ResNet18 cũng cho thấy ít lỗi hơn và khả năng nhận diện tốt hơn cho phần lớn các lớp.

Kết quả này khẳng định rằng đối với các bài toán phân loại hình ảnh phức tạp với số lượng lớp lớn và sự tương đồng về đặc trưng, các kiến trúc mạng sâu như ResNet với cơ chế skip connections là lựa chọn hiệu quả hơn nhiều so với các mạng tích chập đơn giản.

**3.2. Hạn chế của đề tài**

Mặc dù đã đạt được những kết quả nhất định, đề tài vẫn còn một số hạn chế cần được cải thiện:

* Bộ dữ liệu:
  + Kích thước hạn chế: Dữ liệu hoa gồm 102 lớp có thể vẫn còn tương đối hạn chế về số lượng mẫu trên mỗi lớp, đặc biệt đối với các lớp thiểu số. Điều này gây khó khăn cho mô hình trong việc học đầy đủ các đặc trưng của từng lớp, dẫn đến hiệu suất thấp ở một số lớp cụ thể.
  + Đa dạng hình ảnh: Có thể dữ liệu chưa đủ đa dạng về điều kiện chụp (góc độ, ánh sáng, nền), dẫn đến việc mô hình chưa thực sự mạnh mẽ trước các biến thể trong thực tế.
* Kiến trúc mô hình:
  + Mô hình Custom ResNet18: Mặc dù đã tự xây dựng kiến trúc ResNet18, việc tối ưu hóa sâu hơn về kiến trúc (ví dụ: số lượng filter, kích thước kernel, các block) và các siêu tham số (learning rate scheduler, batch size, weight decay) có thể chưa được thực hiện triệt để do giới hạn về thời gian và tài nguyên tính toán.
  + Chưa sử dụng các mô hình tiền huấn luyện (Pre-trained models): Việc sử dụng các mô hình tiên tiến như ResNet18, ResNet50, VGG, Inception đã được tiền huấn luyện trên các bộ dữ liệu lớn như ImageNet, sau đó tinh chỉnh (fine-tuning) cho bài toán cụ thể, thường mang lại hiệu suất cao hơn nhiều và hội tụ nhanh hơn. Đề tài chưa áp dụng phương pháp này.
* Hiệu suất: Độ chính xác 73.59% của ResNet18 tuy đã tốt hơn SimpleCNN nhưng vẫn chưa đạt mức "sẵn sàng ứng dụng" trong nhiều trường hợp thực tế, đặc biệt khi có những lớp bị phân loại kém.
* Tài nguyên và thời gian: Việc huấn luyện các mạng sâu đòi hỏi tài nguyên tính toán lớn (GPU) và thời gian dài, điều này là một hạn chế trong quá trình nghiên cứu và tối ưu.

**3.3. Định hướng mở rộng**

Để khắc phục các hạn chế và nâng cao hơn nữa hiệu quả của đề tài, các định hướng mở rộng sau đây có thể được xem xét:

* Tăng cường dữ liệu (Data Augmentation): Áp dụng các kỹ thuật tăng cường dữ liệu phức tạp hơn như xoay, lật, cắt ngẫu nhiên, thay đổi độ sáng/tương phản, thêm nhiễu, hoặc sử dụng các thư viện như Albumentations để tạo ra nhiều biến thể của ảnh huấn luyện, giúp mô hình học được các đặc trưng mạnh mẽ và khái quát hóa tốt hơn, đồng thời giảm thiểu overfitting.
* Sử dụng mô hình tiền huấn luyện (Pre-trained Models) và Fine-tuning: Đây là định hướng quan trọng nhất để cải thiện hiệu suất. Thay vì huấn luyện từ đầu, có thể tải các mô hình như ResNet18, ResNet50, EfficientNet, Vision Transformer đã được tiền huấn luyện trên bộ dữ liệu ImageNet khổng lồ. Sau đó, chỉ cần tinh chỉnh (fine-tuning) các lớp cuối cùng hoặc toàn bộ mô hình trên tập dữ liệu hoa của mình. Điều này thường mang lại độ chính xác cao hơn đáng kể và hội tụ nhanh hơn.
* Tối ưu hóa siêu tham số (Hyperparameter Tuning): Sử dụng các kỹ thuật tự động tìm kiếm siêu tham số như Grid Search, Random Search, hoặc các thư viện tối ưu hóa nâng cao như Optuna, Weights & Biases để tìm ra bộ siêu tham số (learning rate, batch size, optimizer, weight decay, v.v.) tối ưu nhất cho mô hình ResNet18.
* Cải thiện kiến trúc: Thử nghiệm với các biến thể của kiến trúc ResNet (ví dụ: ResNet34, ResNet50) hoặc các kiến trúc hiện đại khác (như EfficientNet, Vision Transformer) để tìm ra mô hình phù hợp nhất cho bài toán.
* Kỹ thuật Regularization nâng cao: Áp dụng các kỹ thuật regularization khác ngoài Dropout và Batch Normalization, ví dụ như Label Smoothing, Mixup, CutMix để tăng khả năng khái quát hóa và giảm overfitting.
* Xử lý dữ liệu mất cân bằng lớp (Imbalanced Data Handling): Nếu bộ dữ liệu có sự chênh lệch lớn về số lượng mẫu giữa các lớp, có thể áp dụng các kỹ thuật như lấy mẫu quá mức (oversampling) cho lớp thiểu số (ví dụ: SMOTE), lấy mẫu dưới mức (undersampling) cho lớp đa số, hoặc sử dụng hàm mất mát có trọng số (weighted loss function) để cân bằng sự đóng góp của các lớp.
* Đánh giá trên tập kiểm thử (Test Set): Để có cái nhìn khách quan nhất về hiệu suất cuối cùng của mô hình, cần có một tập dữ liệu kiểm thử độc lập, không được sử dụng trong quá trình huấn luyện và validation, để đánh giá duy nhất một lần sau khi mô hình được chọn.
* Phân tích lỗi sâu hơn: Nghiên cứu kỹ hơn các trường hợp mà mô hình dự đoán sai (thông qua ma trận nhầm lẫn và trực quan hóa ảnh) để hiểu rõ lý do và tìm cách cải thiện, ví dụ: các loài hoa dễ bị nhầm lẫn với nhau nhất.

**Tài liệu tham khảo**

<https://www.geeksforgeeks.org/resnet18-from-scratch-using-pytorch/>

<https://aicandy.vn/mo-hinh-resnet-dot-pha-trong-nhan-dien-hinh-anh/>

<https://pbcquoc.github.io/cnn/>

<https://bizfly.vn/techblog/convolutional-neural-network.html>

<https://ichi.pro/vi/tong-quan-ve-resnet-va-cac-bien-the-cua-no-90717811990581>

<https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2>

<https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>