

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH
HỌC KÌ 1, NĂM HỌC 2023-2024

Tên đề tài: Phân loại ảnh dựa trên kỹ thuật học sâu

Giảng viên hướng dẫn:

Họ và tên: Nguyễn Mộng Hiền

Sinh viên thực hiện:

Họ và tên: Võ Anh Duy

MSSV: 110120167

Lớp: DA20TTA

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH
HỌC KÌ 1, NĂM HỌC 2023-2024

Tên đề tài: Phân loại ảnh dựa trên kỹ thuật học sâu

Giảng viên hướng dẫn:

Họ và tên: Nguyễn Mộng Hiền

Sinh viên thực hiện:

Họ và tên: Võ Anh Duy

MSSV: 110120167

Lớp: DA20TTA

[illegible]

(Ký và ghi rõ họ tên)

[illegible]

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Tôi xin được gửi đến quý thầy cô ở Khoa Kỹ thuật và Công nghệ vì những kiến thức đã được học tập từ quý thầy cô, cũng như các kiến thức trong quá trình tự tìm hiểu và đặc biệt với sự hướng dẫn nhiệt tình của thầy Nguyễn Mộng Hiền, tôi đã hoàn thành được đề tài “Phân loại ảnh dựa trên kỹ thuật học sâu”. Mong nhận được sự đóng góp ý kiến của quý thầy cô.

Tôi xin chân thành cảm ơn.

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

1. Vấn đề nghiên cứu:

- Nghiên cứu về python và neural network.
- Chuẩn bị dữ liệu về hình ảnh cụ thể.
- Cài đặt các thư viện liên quan đến neural network để phân loại hình ảnh cụ thể.

2. Hướng tiếp cận:

- Hiểu cách xây dựng mô hình phân loại ảnh dựa trên neural network.
- Xây dựng phần mềm phân loại ảnh.
- Cài đặt các thư viện cần thiết cho phân loại ảnh(python).

3. Cách giải quyết:

- Nghiên cứu tài liệu về neural network và python.
- Nghiên cứu cách phân loại ảnh trong python dựa vào kỹ thuật học sâu.

4. Kết quả đạt được:

Hoàn thành các nội dung về phân loại ảnh dựa vào kỹ thuật học sâu.

MỞ ĐẦU

1. Lý do chọn đề tài:

Ngày nay, do có nhiều loại ảnh cần được phân loại trên máy tính để nắm rõ chi tiết cũng như đặc điểm của một đối tượng cụ thể. Nên có rất nhiều nơi nghiên cứu về các thuật toán cũng như mô hình hoạt động của việc phân loại ảnh. Để có thể lọc ra ảnh của một đối tượng cụ thể trong hệ thống thì cần phải có một kỹ thuật phân loại ảnh cụ thể. Vì vậy, đó chính là lí do cần phải “Phân loại ảnh dựa trên kỹ thuật học sâu” nhằm phục vụ cho việc phân loại một đối tượng trong ảnh một cách nhanh chóng.

2. Mục đích nghiên cứu:

- Nghiên cứu tài liệu về: Phân loại ảnh dựa trên kỹ thuật học sâu, công cụ thực hiện: Python.
- Thực nghiệm: cài đặt và kiểm thử dữ liệu.

3. Đối tượng: mạng neural network.

4. Phạm vi nghiên cứu: Phân loại ảnh với tập dữ liệu thử nghiệm của một đối tượng dựa trên neural network

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN	1
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT	2
2.1. Cơ sở lý thuyết:	2
2.1.1. Tổng quan về phân loại ảnh dựa trên kỹ thuật học sâu:[1]	2
2.1.2. Tổng quan về các loại môi trường dùng để phân loại ảnh trong python:..	2
2.1.3. Tổng quan về neural network và mô hình hoạt động:	4
2.1.4. Tổng quan về python và các công cụ(thư viện) phân loại ảnh:	5
2.1.5. Tổng quan về python và các công cụ(thư viện) phân loại ảnh:[3]	6
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	8
3.1. Mô tả bài toán:	8
3.2. Phương pháp đề xuất: Sử dụng phương pháp Mạng Neural Convolutional (CNN) Tiên Tiến.	9
3.3. Sơ đồ khối:	9
3.4. Cách sử dụng sơ đồ khối:	10
3.5. Cách cài đặt python:[6]	10
3.6. Hiện thực hóa phương pháp đề xuất và hướng dẫn sử dụng:	13
3.6.1. Cài đặt các thư viện:	13
3.6.2. Tải dữ liệu:	13
3.6.3. Đếm số lượng bộ dữ liệu:	13
3.6.4. Hiển thị hình ảnh đối tượng:	14
3.6.5. Tạo tập dữ liệu cho việc đào tạo(training) và xác thực(validation):[4].	15
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU	29
Kết quả đạt được:	29
Hạn chế:	29
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	30
DANH MỤC TÀI LIỆU THAM KHẢO	31
1. Nội dung đề tài:	1
2. Đối tượng nghiên cứu: Ảnh của một đối tượng cụ thể.....	1
3. Phương pháp nghiên cứu:	1
3.1. Nghiên cứu lý thuyết:	1
3.2. Nghiên cứu thực nghiệm:	1

4. Phạm vi nghiên cứu: phân loại ảnh với tập dữ liệu thử nghiệm của một đối tượng dựa vào neural network

5. Phương pháp thực hiện:.....

6. Yêu cầu:

6.1. Thực hiện phân loại ảnh trên hệ thống:.....

7. Kết quả đạt được

2

2

2

2

2

DANH MỤC HÌNH ẢNH

Hình 2.1. Giao diện Jupyter notebook.....	3
Hình 2.2: Giao diện google colab.....	3
Hình 2.3: Giao diện vscode.....	4
Hình 2.4: Kiến trúc mạng nơ-ron.....	5
Hình 2.5: Quá trình phân loại ảnh của CNNs.....	7
Hình 2.6: Phân loại ảnh bằng CNNs.....	7
Hình 3.1: Sơ đồ khối phân loại ảnh.....	9
Hình 3.2: Trang web cài đặt python.....	11
Hình 3.3: chạy file python-3.11.6.exe.....	11
Hình 3.4: Cài đặt python.....	11
Hình 3.5: Tiến trình cài đặt.....	12
Hình 3.6: Giao diện sau khi cài đặt thành công.....	12
Hình 3.7: Tải thành công bộ dữ liệu.....	13
Hình 3.8: Ảnh của loài hoa trong lớp rose.....	14
Hình 3.9: Ảnh của loài hoa trong lớp tulips.....	15
Hình 3.10: Kết quả sau khi huấn luyện.....	16
Hình 3.11: Kết quả sau khi xác thực.....	16
Hình 3.12: Các lớp của loài hoa.....	16
Hình 3.13: Kết quả trực quan hóa dữ liệu.....	17
Hình 3.14: Kết quả chuẩn hóa dữ liệu.....	18
Hình 3.15: Kết quả tóm tắt mô hình.....	20
Hình 3.16: Kết quả đào tạo mô hình.....	21
Hình 3.17: Kết quả đào tạo.....	22
Hình 3.18: Ảnh chụp nhiều góc độ sau khi làm tăng dữ liệu.....	23
Hình 3.19: Đoạn code lọc bỏ dữ liệu.....	24
Hình 3.20: Kết quả đào tạo lại mô hình sau khi lọc bỏ dữ liệu.....	25
Hình 3.21: Kết quả đào tạo.....	26
Hình 3.22: Kết quả đạt được độ tin cậy chính xác của đối tượng.....	27
Hình 3.23: Phân loại 2 loài hoa daisy và tulips.....	27
Hình 3.24: Thực hiện load ảnh và đọc file ảnh đã load.....	28

CHƯƠNG 1: TỔNG QUAN

Ngày nay, công nghệ 4.0 đang phát triển mạnh mẽ nên nhiều nơi hầu hết đều phân loại ảnh trên máy tính bằng kỹ thuật học sâu. Việc phân loại ảnh dựa trên kỹ thuật học sâu có ưu và nhược điểm sau:

Ưu điểm:

- Giúp cho việc phân loại đối tượng trong ảnh trở nên nhanh chóng.
- Giúp tìm ra được độ tin cậy để quyết định chính xác của đối tượng trong ảnh

Nhược điểm: Do có nhiều loại ảnh của một đối tượng nên việc phân loại ảnh dựa trên kỹ thuật học sâu là một ưu điểm

Do hình thức phân loại bằng cách sử dụng kỹ thuật này, ở nhiều nơi đã trải qua nhiều năm cùng việc cải tiến chức năng của hệ thống cũng như kỹ thuật phân loại ảnh.

Để có thể phân loại ảnh một cách dễ dàng cần phải có một phần mềm cụ thể trên máy tính. Việc xây dựng hệ thống phân loại ảnh này sẽ giúp nhiều nơi cần sử dụng phân loại ảnh để phân biệt đối tượng dễ dàng hơn.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 . Cơ sở lý thuyết:

2.1.1 . Tổng quan về phân loại ảnh dựa trên kỹ thuật học sâu:[1]

Phân loại ảnh dựa trên kỹ thuật học sâu là một lĩnh vực nghiên cứu trong lĩnh vực thị giác máy tính, mà mục tiêu chính là sử dụng kỹ thuật học máy, đặc biệt là neural network để tự động phân loại và nhận biết đối tượng đặc trưng trong hình ảnh.

Môi trường thực hiện phân loại ảnh phổ biến hiện nay: Jupyter Notebooks, Google Colab, Visual Studio Code,...

Google colab là dịch vụ Jupyter Notebooks miễn phí được cung cấp bởi google. Đây là môi trường cơ bản dùng để thực thi các câu lệnh trong python. GPU và TPU trong google colab cung cấp cho người dùng các khả năng:

Tăng tốc huấn luyện mô hình: xử lý đồng thời và tính toán số học lớn, giúp tăng tốc quá trình huấn luyện mô hình học sâu.

Xử lý dữ liệu lớn: giúp xử lý các tập dữ liệu có kích thước lớn, giảm thời gian và tài nguyên cần thiết cho việc đào tạo mô hình.

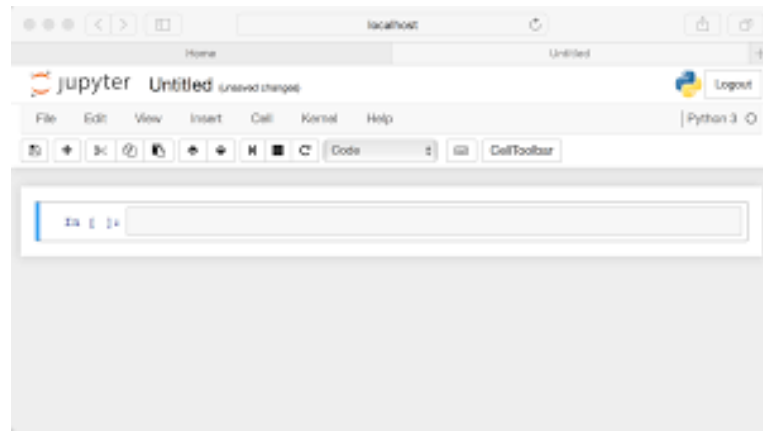
Hỗ trợ nhanh chóng cho thử nghiệm mô hình: cho phép người dùng thử nghiệm và điều chỉnh mô hình một cách nhanh chóng. Điều này làm tăng hiệu suất trong quá trình phát triển và tối ưu hóa mô hình.

Google colab sử dụng ngôn ngữ lập trình Python. Ngôn ngữ lập trình cho phép người dùng thực nghiệm tập dữ liệu, cập nhật và quản lý tài nguyên trên hệ thống. Tất cả các máy chủ đều có chung một GPU và TPU.

2.1.2 . Tổng quan về các loại môi trường dùng để phân loại ảnh trong python:

2.1.2.1 . Jupyter Notebook:

Jupyter Notebook là một nền tảng tính toán khoa học mã nguồn mở, với khả năng nổi bật cho phép tương tác trực tiếp với từng dòng code (interactive), hỗ trợ hơn 40 ngôn ngữ lập trình, trong đó tập trung vào 3 ngôn ngữ là Julia, Python và R, và nó cũng là một môi trường miễn phí cho các lập trình viên chuyên dùng deep learning để phân loại đối tượng trong ảnh.



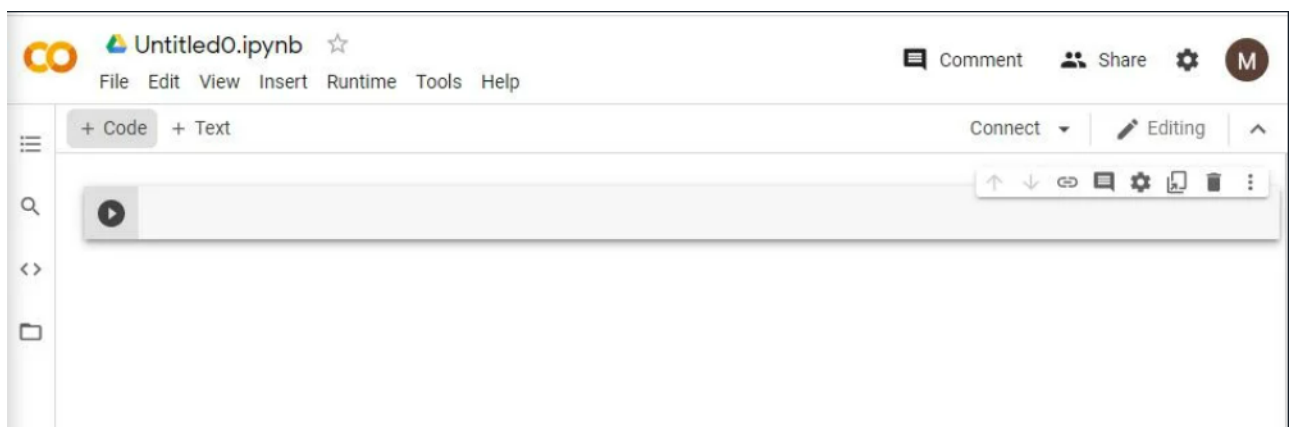
Hình 2.1. Giao diện Jupyter notebook

2.1.2.2. Google colab:

- Google colab là một môi trường cho phép thực thi lệnh python trên nền tảng đám mây mà không cần cài đặt về máy, mọi câu lệnh có thể chạy thông qua trình duyệt, cung cấp tài nguyên CPU tốc độ cao cho đến GPUs và TPUs đều có thể thực hiện chạy chương trình.

- Lí do nên sử dụng google colab:

- + Các thư viện được cài đặt sẵn;
- + Được lưu trên đám mây;
- + Sử dụng GPU và TPU miễn phí.



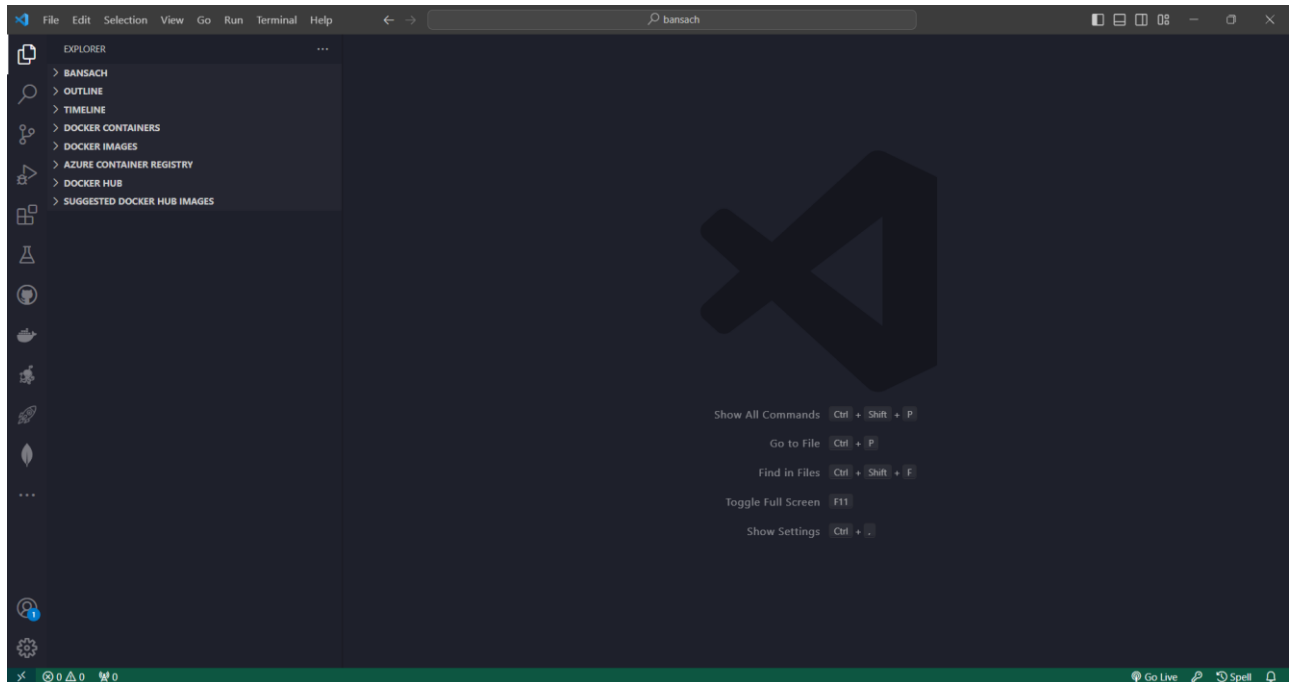
Hình 2.2: Giao diện google colab

2.1.2.3. Visual Studio Code(VS Code):

- VS Code là một trình soạn thảo mã nguồn mở miễn phí. Đây là một môi trường phát triển tích hợp(IDE) mạnh mẽ với nhiều tính năng và tiện ích hỗ trợ cho các nhà phát triển phần mềm.

- Ưu điểm:

- + Đa dạng ngôn ngữ lập trình;
- + Các tiện ích mở rộng rất đa dạng và phong phú;
- + Tích hợp các tính năng quan trọng như bảo mật(Git), khả năng tăng tốc xử lý vòng lặp(Debug)...



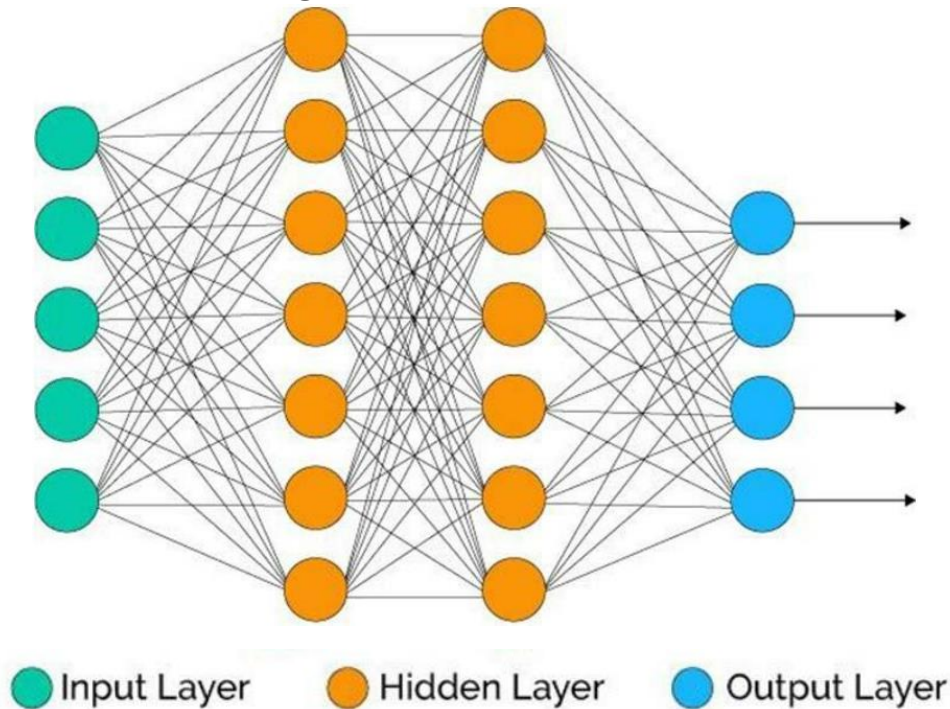
Hình 2.3: Giao diện vscode

2.1.3 . Tổng quan về neural network và mô hình hoạt động:

2.1.3.1. Tổng quan neural network:

Neural network(Mạng nơ-ron) là một phương thức trong lĩnh vực trí tuệ nhân tạo, được sử dụng để dạy máy tính xử lý dữ liệu theo cách được lấy cảm hứng từ bộ não con người. Đây là một loại quy trình máy học, được gọi là deep learning, sử dụng các nút hoặc nơ-ron liên kết với nhau trong một cấu trúc phân lớp tương tự như bộ não con người. Vì vậy, mạng nơ-ron nhân tạo nhằm tới giải quyết các vấn đề phức tạp, chẳng hạn như tóm tắt tài liệu hoặc nhận diện khuôn mặt, với độ chính xác cao hơn.

2.1.3.2. Kiến trúc mạng nơ-ron:



Hình 2.4: Kiến trúc mạng nơ-ron

- Tầng input layer (tầng vào): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
- Tầng hidden layer (tầng ẩn): Tầng này nằm giữa tầng vào và tầng ra nó thể hiện cho quá trình suy luận logic của mạng.
- Tầng output layer (tầng ra): Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.

2.1.3.3. Cách thức hoạt động của mô hình:

Đầu tiên, mô hình sẽ xác định ảnh của một đối tượng ở input layer, tiếp theo thì mô hình sẽ phân tích đặc điểm của đối tượng bằng thuật toán ở hidden layer, sau khi đã phân tích xong thì mô hình sẽ kết luận đối tượng đặc trưng của ảnh trong ảnh.

2.1.4 . Tổng quan về python và các công cụ(thư viện) phân loại ảnh:

2.1.4.1. Python:

- Python là một ngôn ngữ lập trình thông dịch và bậc cao do Guido van Rossum phát triển vào năm 1980.

- Ưu và nhược điểm của python:

- * Ưu điểm:

- + Dễ học và dễ đọc;
- + Tránh tác hại từ lỗi phần mềm;
- + Khả năng ứng dụng rộng rãi.

* Nhược điểm:

- + Tiêu thụ bộ nhớ lớn;
- + Không thích hợp cho phát triển trò chơi và thiết bị di động;
- + Khó kiểm tra.

2.1.4.2. Công cụ(thư viện) phân loại ảnh:[2]

- Tensorflow: là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

- Pillow là một fork từ thư viện PIL của Python được sử dụng để xử lý hình ảnh. So với PIL thì Pillow được cập nhật thường xuyên và đánh giá cao hơn. (PIL đã không được cập nhật từ năm 2009).

- Numpy là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.

- Matplotlib là một thư viện dùng để trực quan hóa biểu đồ trong python và Nó có thể mang đến những công cụ phục vụ việc vẽ những biểu đồ điểm, đường, cột, hay biểu đồ tròn...

- Module os trong Python cung cấp các chức năng được sử dụng để tương tác với hệ điều hành và cũng có được thông tin liên quan về nó.

2.1.5. Tổng quan về python và các công cụ(thư viện) phân loại ảnh:[3]

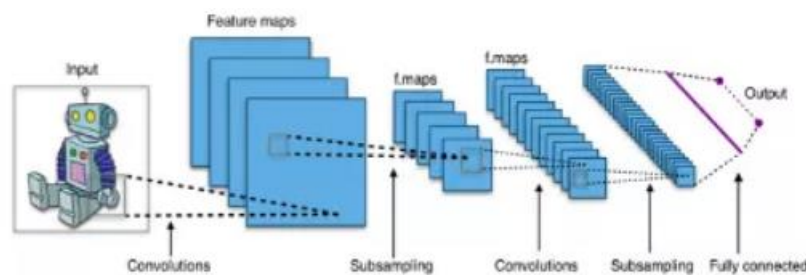
2.1.5.1. CNN(Convolutional Neural Network):

- Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

- Cấu trúc:

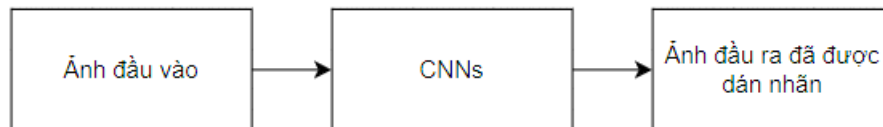
+ Mạng CNN là một tập hợp các lớp tích chập chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node.

+ Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer). Còn trong mô hình CNNs thì ngược lại. Các lớp liên kết được với nhau thông qua cơ chế tích chập. Lớp tiếp theo là kết quả phép tính tích chập từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ.



Hình 2.5: Quá trình phân loại ảnh của CNNs

2.1.5.2. Quá trình phân loại ảnh của mạng neural network:



Hình 2.6: Phân loại ảnh bằng CNNs

- Giải thích mô hình: đầu tiên cho ảnh đầu vào mô hình, sau đó xử lý phân loại bằng mạng neural network bằng các đặc trưng của đối tượng trong ảnh, cuối cùng cho kết quả ảnh đầu ra giống với đặc điểm mà mạng neural network đã phân loại.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 . Mô tả bài toán:

- Phân loại ảnh dựa trên kỹ thuật học sâu là quá trình sử dụng mô hình học sâu để tự động phân loại hình ảnh thành các lớp hoặc nhãn khác nhau dựa trên đặc điểm và thông tin trong ảnh.

- Quá trình phân loại ảnh thông qua học sâu thường bao gồm các bước chính sau:

- Thu thập và Chuẩn bị Dữ liệu:

+ Thu thập dữ liệu ảnh: Dữ liệu gồm các hình ảnh đã được gán nhãn theo từng lớp mong muốn.

- Xây dựng Mô hình Học Sâu:

+ Sử dụng các kiến trúc mô hình học sâu như Convolutional Neural Networks (CNNs) thường được sử dụng cho việc phân loại ảnh.

+ Quá trình huấn luyện mô hình trên dữ liệu đã chuẩn bị: Mô hình học từ dữ liệu thông qua việc điều chỉnh trọng số (weights) để phát hiện và học các đặc trưng của từng lớp ảnh.

- Đánh giá và Tinh chỉnh Mô hình:

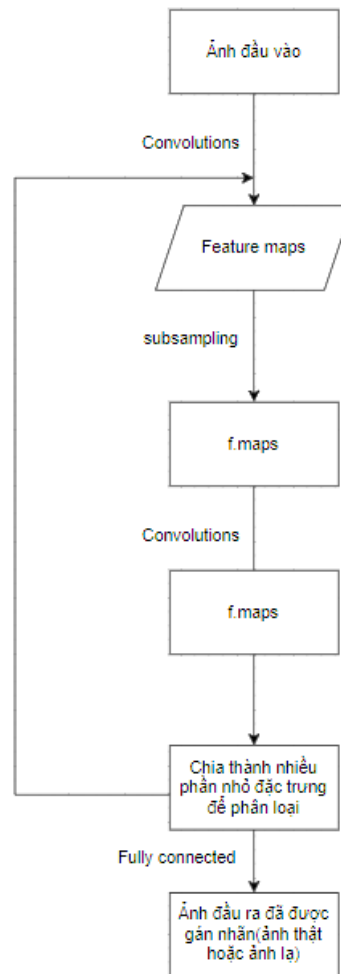
+ Sử dụng tập dữ liệu kiểm tra để đánh giá hiệu suất của mô hình trên các ảnh mới chưa được huấn luyện.

- Dự đoán và Phân loại ảnh mới:

+ Sau khi mô hình đã được huấn luyện và đánh giá, nó có thể được sử dụng để dự đoán và phân loại ảnh mới thành các lớp đã được định nghĩa trước.

3.2 . Phương pháp đề xuất: Sử dụng phương pháp Mạng Neural Convolutional (CNN) Tiên Tiến.

3.3 . Sơ đồ khối:



Hình 3.1: Sơ đồ khối phân loại ảnh

- Giải thích các từ ngữ trong sơ đồ khối:

+ Input(Ảnh đầu vào) là đầu vào của mạng nơ-ron, trong trường hợp này là một hình ảnh. Hình ảnh được biểu diễn dưới dạng một mảng các pixel, mỗi pixel có một giá trị cường độ màu.

+ Feature maps là kết quả của quá trình tích chập. Tích chập là một phép toán toán học được sử dụng để trích xuất các đặc trưng từ dữ liệu. Trong trường hợp của CNN, tích chập được sử dụng để trích xuất các đặc trưng từ hình ảnh.

+ Subsampling là quá trình giảm kích thước của feature maps. Subsampling thường được thực hiện bằng cách sử dụng phép chia lấy trung bình.

+ Fully connected layers là các lớp nơ-ron có các kết nối giữa tất cả các nơ-ron trong một lớp. Các lớp này được sử dụng để thực hiện phân loại hoặc các nhiệm vụ xử lý ngôn ngữ tự nhiên khác.

+ Convolution là một phép toán toán học được sử dụng để trích xuất các đặc trưng từ dữ liệu. Trong trường hợp của CNN, tích chập được sử dụng để trích xuất các đặc trưng từ hình ảnh.

3.4 . Cách sử dụng sơ đồ khối:

Bước 1: Đầu vào của mạng nơ-ron là một hình ảnh. Hình ảnh được biểu diễn dưới dạng một mảng các pixel, mỗi pixel có một giá trị cường độ màu.

Bước 2: Quá trình tích chập được sử dụng để trích xuất các đặc trưng từ hình ảnh. Tích chập được thực hiện bằng cách sử dụng một bộ lọc (filter). Bộ lọc là một mảng các số có kích thước nhỏ hơn kích thước của hình ảnh. Kết quả của phép toán này được lưu trữ trong một feature map.

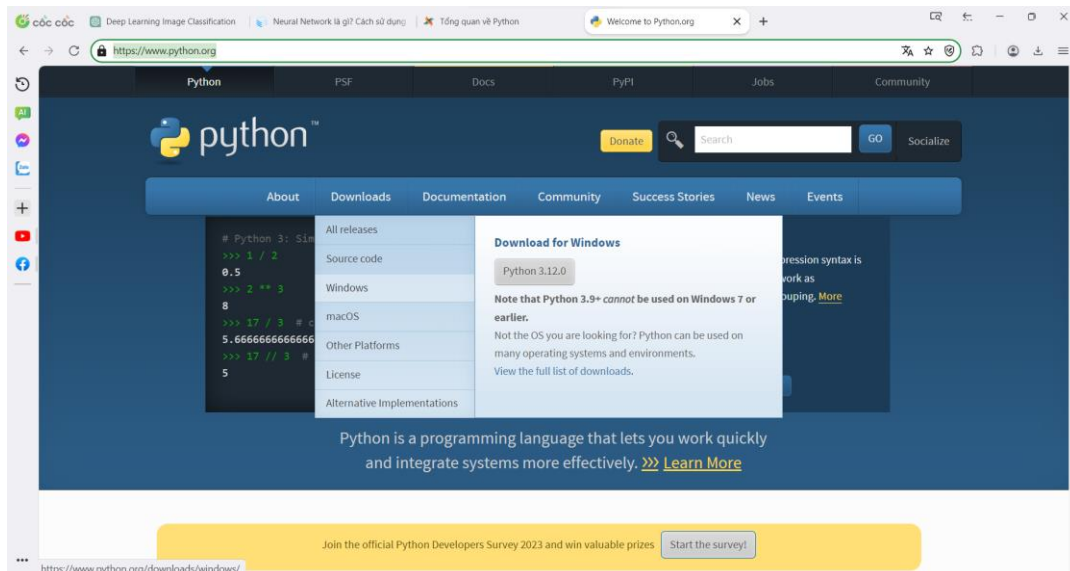
Bước 3: Quá trình subsampling được sử dụng để giảm kích thước của feature maps. Subsampling thường được thực hiện bằng cách sử dụng phép chia lấy trung bình. Subsampling giúp giảm số lượng các tham số cần thiết cho mạng nơ-ron, đồng thời cũng giúp tăng khả năng chống nhiễu của mạng.

Bước 4: Các feature maps được kết nối với các lớp fully connected. Các lớp này được sử dụng để thực hiện phân loại hoặc các nhiệm vụ xử lý ngôn ngữ tự nhiên khác.

3.5 . Cách cài đặt python:[6]

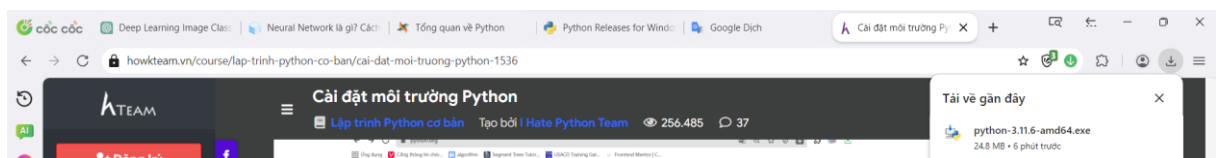
- Cách cài đặt:

+ Bước 1: vào trang web <https://www.python.org>, sau đó tiến hành chọn phiên bản muốn cài đặt.



Hình 3.2: Trang web cài đặt python

+ Bước 2: Sau download hoàn tất. Chúng ta nhấn chọn chạy file python-3.11.6.exe để bắt đầu tiến trình cài đặt.



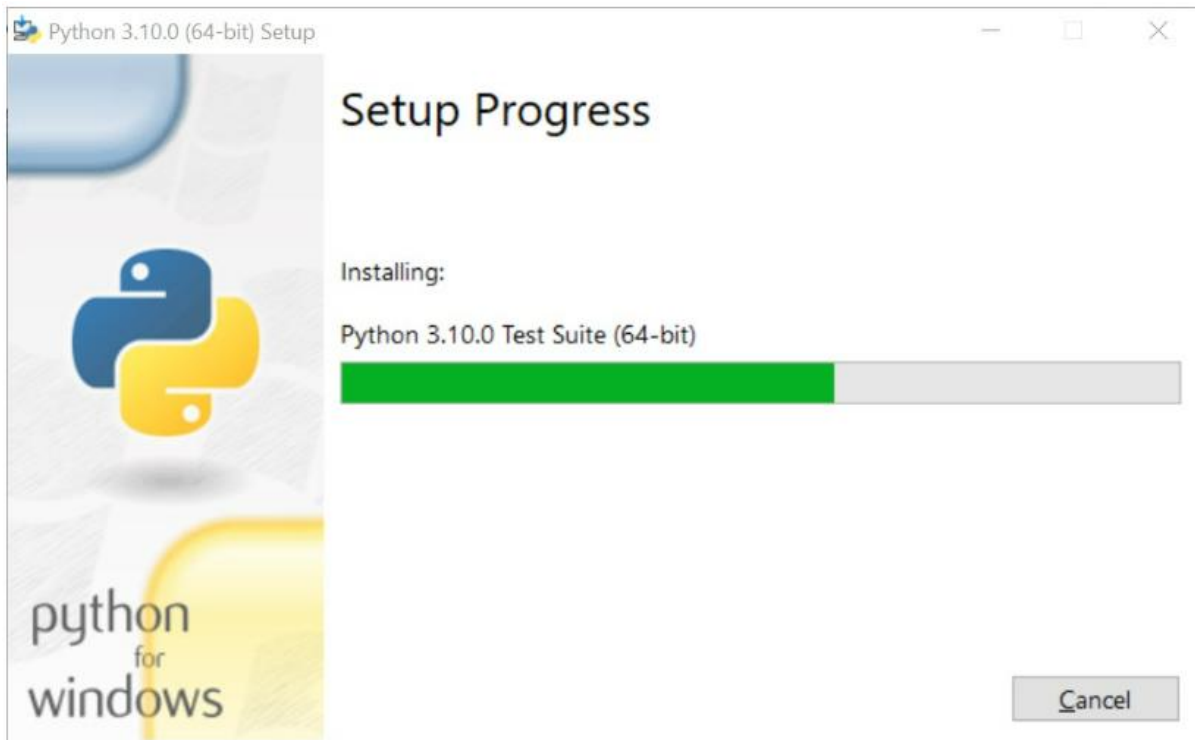
Hình 3.3: chạy file python-3.11.6.exe

+ Bước 3: Tick vào ô Add python 3.11.6 to PATH và chọn Install Now

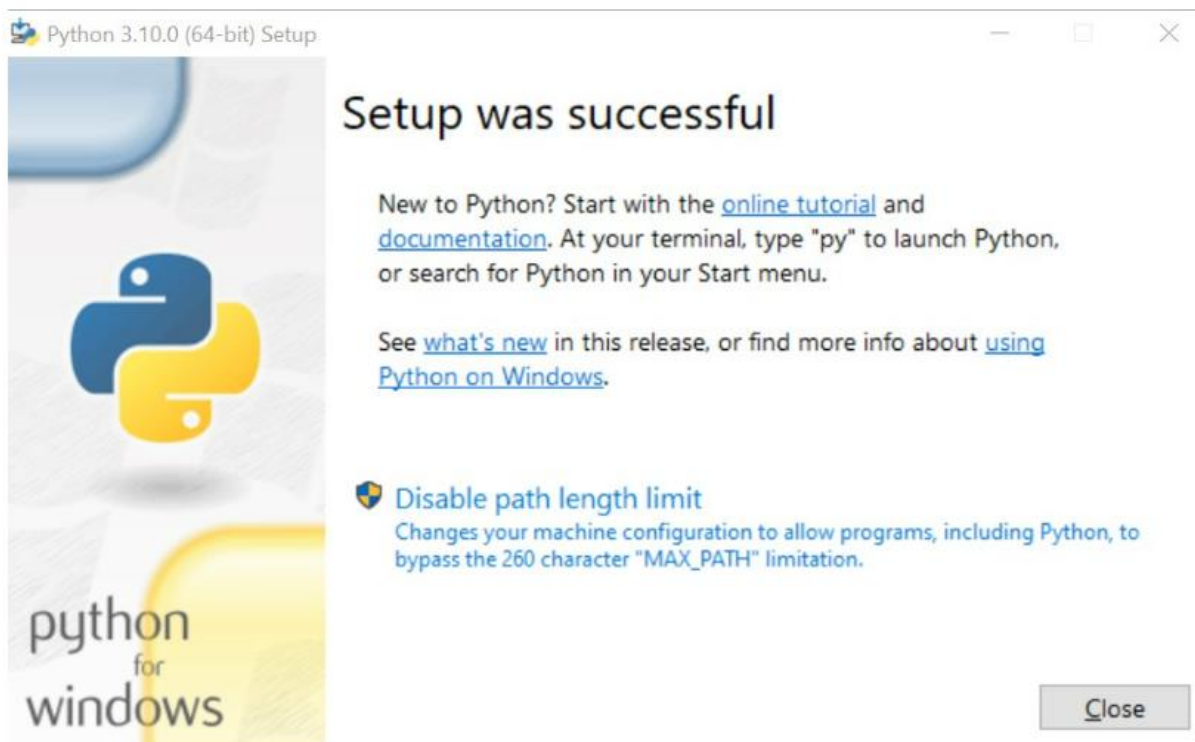


Hình 3.4: Cài đặt python

+ Bước 4: Khi cửa sổ hiển thị Setup was successful là ta đã cài đặt thành công môi trường python > Close



Hình 3.5: Tiến trình cài đặt



Hình 3.6: Giao diện sau khi cài đặt thành công

3.6 . Hiện thực hóa phương pháp đề xuất và hướng dẫn sử dụng:

3.6.1 . Cài đặt các thư viện:

```
import matplotlib.pyplot as plt

import numpy as np

import os

import PIL

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential
```

3.6.2 . Tải dữ liệu:

```
import pathlib

dataset_url="https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"

data_dir=tf.keras.utils.get_file('flower_photos',origin=dataset_url,

untar=True)

data_dir = pathlib.Path(data_dir)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz
228813984/228813984 [=====] - 2s 0us/step
```

Hình 3.7: Tải thành công bộ dữ liệu

3.6.3 . Đếm số lượng bộ dữ liệu:

```
image_count = len(list(data_dir.glob('*/*.jpg')))

print(image_count)
```

3.6.4 . Hiện thị hình ảnh đối tượng:

```
▶ roses = list(data_dir.glob('roses/*'))  
PIL.Image.open(str(roses[0]))
```



```
[5] PIL.Image.open(str(roses[1]))
```



Hình 3.8: Ảnh của loài hoa trong lớp rose


```
[6] tulips = list(data_dir.glob('tulips/*'))  
PIL.Image.open(str(tulips[0]))
```



```
PIL.Image.open(str(tulips[1]))
```



Hình 3.9: Ảnh của loài hoa trong lớp tulips

3.6.5 . Tạo tập dữ liệu cho việc đào tạo(training) và xác thực(validation):[4]

Bước 1: thiết lập số lượng ảnh cần huấn luyện, chiều dài và chiều rộng của ảnh:

```
batch_size = 32
```

```
img_height = 180
```

```
img_width = 180
```

Bước 2: Tiến hành huấn luyện và xác thực:

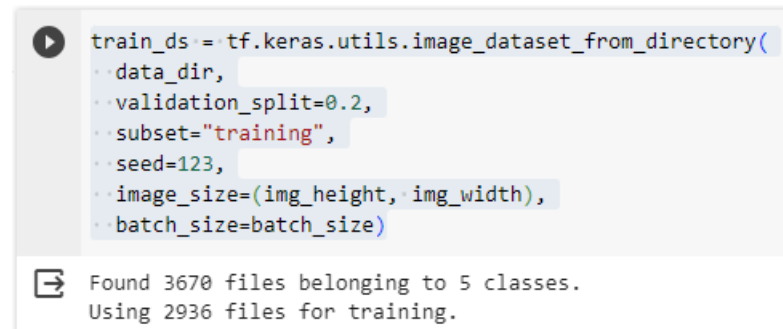
```
train_ds = tf.keras.utils.image_dataset_from_directory(  
data_dir,
```

```
validation_split=0.2,
```

```
subset="training",
```

```
seed=123,
```

```
image_size=(img_height, img_width),  
batch_size=batch_size)
```

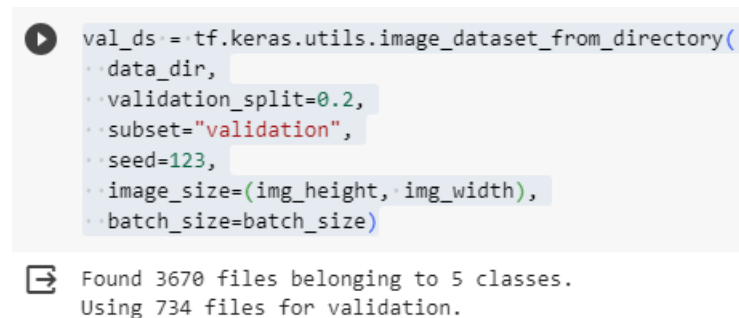


```
train_ds = tf.keras.utils.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    subset="training",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size)
```

Found 3670 files belonging to 5 classes.
Using 2936 files for training.

Hình 3.10: Kết quả sau khi huấn luyện

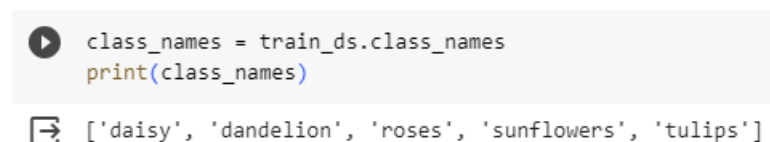
```
val_ds = tf.keras.utils.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    subset="validation",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size)
```



```
val_ds = tf.keras.utils.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    subset="validation",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size)
```

Found 3670 files belonging to 5 classes.
Using 734 files for validation.

Hình 3.11: Kết quả sau khi xác thực



```
class_names = train_ds.class_names  
print(class_names)
```

['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

Hình 3.12: Các lớp của loài hoa

Bước 3: Trực quan hóa dữ liệu:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))

for images, labels in train_ds.take(1):

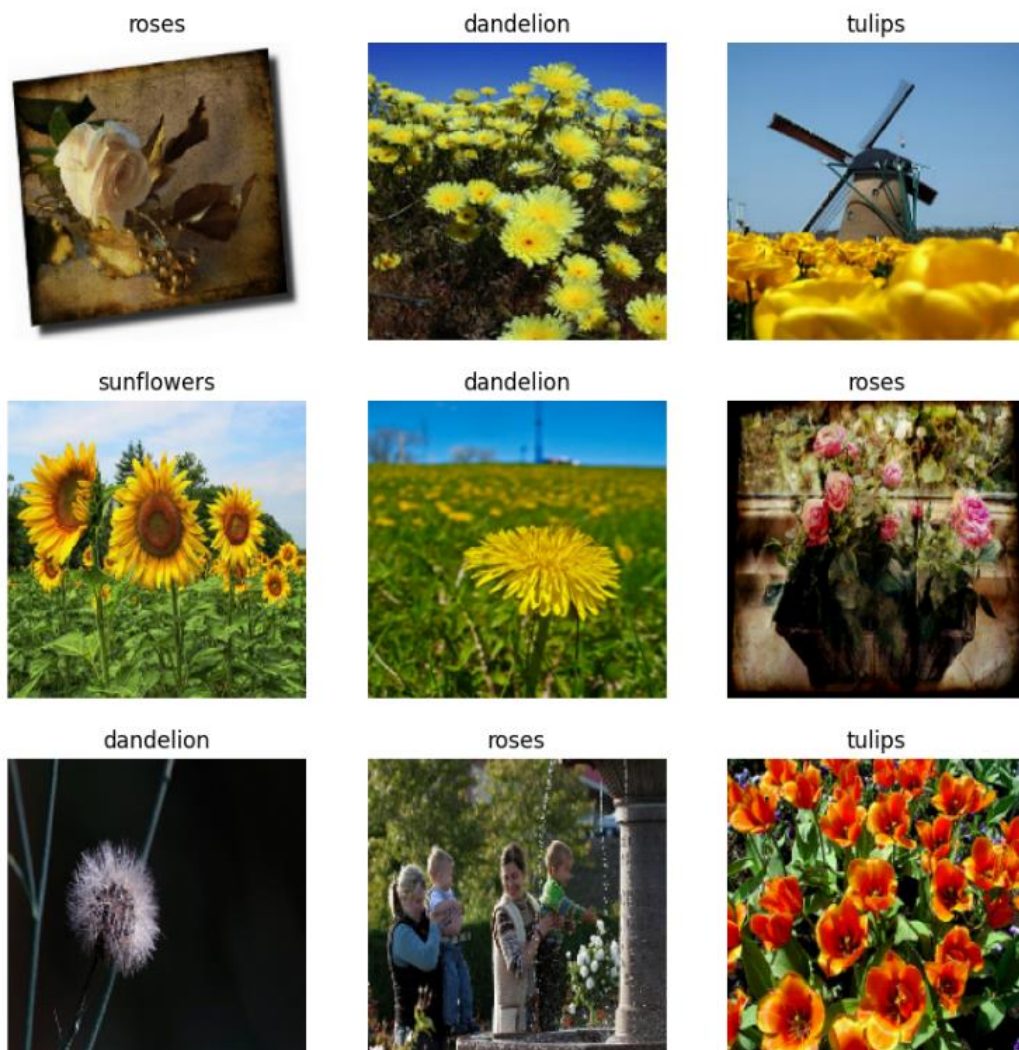
    for i in range(9):

        ax = plt.subplot(3, 3, i + 1)

        plt.imshow(images[i].numpy().astype("uint8"))

        plt.title(class_names[labels[i]])

        plt.axis("off")
```



Hình 3.13: Kết quả trực quan hóa dữ liệu

Bước 4: Dùng câu lệnh in ra hình dạng tenxơ của ảnh. Đây là một khối gồm 32 hình ảnh có kích thước 180x180x3(số cuối dùng là đề cập đến các kênh màu RGB). label_batch là một tenxơ của hình (32,) , đây là những nhãn tương ứng với 32 hình ảnh:

```
for image_batch, labels_batch in train_ds:

    print(image_batch.shape)

    print(labels_batch.shape)

    break
```

Bước 5: Định cấu hình cho hiệu suất training và val:

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)

val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Bước 6: Chuẩn hóa dữ liệu:

```
normalization_layer = layers.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))

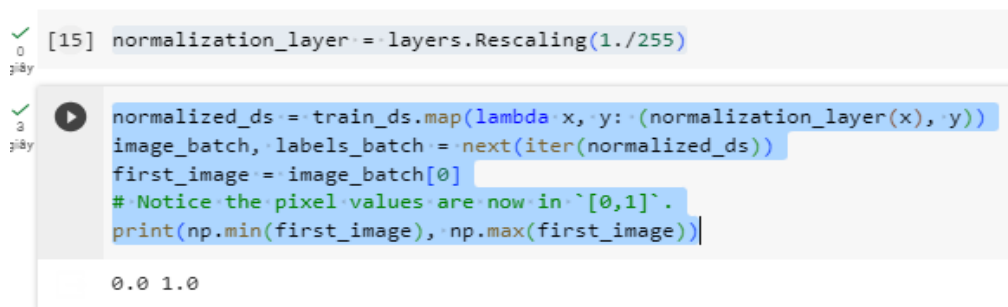
image_batch, labels_batch = next(iter(normalized_ds))

first_image = image_batch[0]

# Notice the pixel values are now in `[0,1]`.

print(np.min(first_image), np.max(first_image))
```

▼ Chuẩn hóa dữ liệu



Hình 3.14: Kết quả chuẩn hóa dữ liệu

Bước 7: Tạo mô hình:

```
num_classes = len(class_names)

model = Sequential([

    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),

    layers.Conv2D(16, 3, padding='same', activation='relu'),

    layers.MaxPooling2D(),

    layers.Conv2D(32, 3, padding='same', activation='relu'),

    layers.MaxPooling2D(),

    layers.Conv2D(64, 3, padding='same', activation='relu'),

    layers.MaxPooling2D(),

    layers.Flatten(),

    layers.Dense(128, activation='relu'),

    layers.Dense(num_classes)

])
```

Bước 8: Biên dịch mô hình:


```
model.compile(optimizer='adam',


              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),

              metrics=['accuracy'])
```

Bước 9: Tóm tắt mô hình:

```
model.summary()
```

 `model.summary()`

 Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3965056
dense_1 (Dense)	(None, 5)	645

=====
 Total params: 3989285 (15.22 MB)
 Trainable params: 3989285 (15.22 MB)
 Non-trainable params: 0 (0.00 Byte)

Hình 3.15: Kết quả tóm tắt mô hình

Bước 10: Đào tạo mô hình:

`epochs=10`

```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)
```

```
[ ] epochs=10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)

Epoch 1/10
92/92 [=====] - 14s 36ms/step - loss: 1.3712 - accuracy: 0.4135 - val_loss: 1.2291 - val_accuracy: 0.4918
Epoch 2/10
92/92 [=====] - 2s 21ms/step - loss: 1.0226 - accuracy: 0.5909 - val_loss: 0.9828 - val_accuracy: 0.5995
Epoch 3/10
92/92 [=====] - 2s 23ms/step - loss: 0.8508 - accuracy: 0.6761 - val_loss: 0.9993 - val_accuracy: 0.5926
Epoch 4/10
92/92 [=====] - 2s 21ms/step - loss: 0.6921 - accuracy: 0.7337 - val_loss: 0.9277 - val_accuracy: 0.6213
Epoch 5/10
92/92 [=====] - 2s 21ms/step - loss: 0.5167 - accuracy: 0.8079 - val_loss: 1.0078 - val_accuracy: 0.6253
Epoch 6/10
92/92 [=====] - 2s 21ms/step - loss: 0.3173 - accuracy: 0.8968 - val_loss: 1.1391 - val_accuracy: 0.6199
Epoch 7/10
92/92 [=====] - 2s 20ms/step - loss: 0.2136 - accuracy: 0.9329 - val_loss: 1.3574 - val_accuracy: 0.6213
Epoch 8/10
92/92 [=====] - 2s 22ms/step - loss: 0.1091 - accuracy: 0.9663 - val_loss: 1.5720 - val_accuracy: 0.5981
Epoch 9/10
92/92 [=====] - 2s 23ms/step - loss: 0.0446 - accuracy: 0.9874 - val_loss: 1.8959 - val_accuracy: 0.6267
Epoch 10/10
92/92 [=====] - 2s 21ms/step - loss: 0.0604 - accuracy: 0.9837 - val_loss: 1.7478 - val_accuracy: 0.6322
```

Hình 3.16: Kết quả đào tạo mô hình

Bước 11: Hình dung kết quả đào tạo:

```
acc = history.history['accuracy']
```

```
val_acc = history.history['val_accuracy']
```

```
loss = history.history['loss']
```

```
val_loss = history.history['val_loss']
```

```
epochs_range = range(epochs)
```

```
plt.figure(figsize=(8, 8))
```

```
plt.subplot(1, 2, 1)
```

```
plt.plot(epochs_range, acc, label='Training Accuracy')
```

```
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
```

```
plt.legend(loc='lower right')
```

```
plt.title("Training and Validation Accuracy")
```

```
plt.subplot(1, 2, 2)
```

```
plt.plot(epochs_range, loss, label='Training Loss')
```

```
plt.plot(epochs_range, val_loss, label='Validation Loss')
```

```
plt.legend(loc='upper right')
```

```
plt.title("Training and Validation Loss")
```

```
plt.show()
```

Hình dung kết quả đào tạo

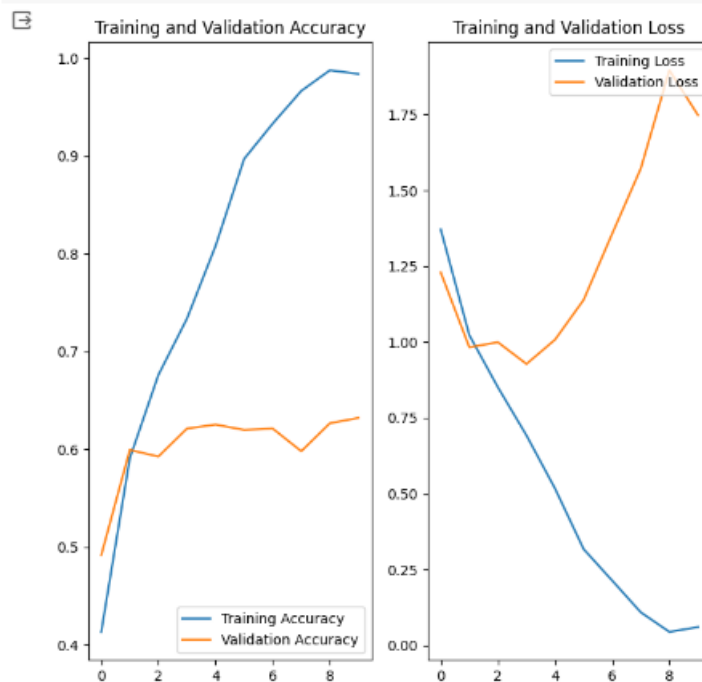
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

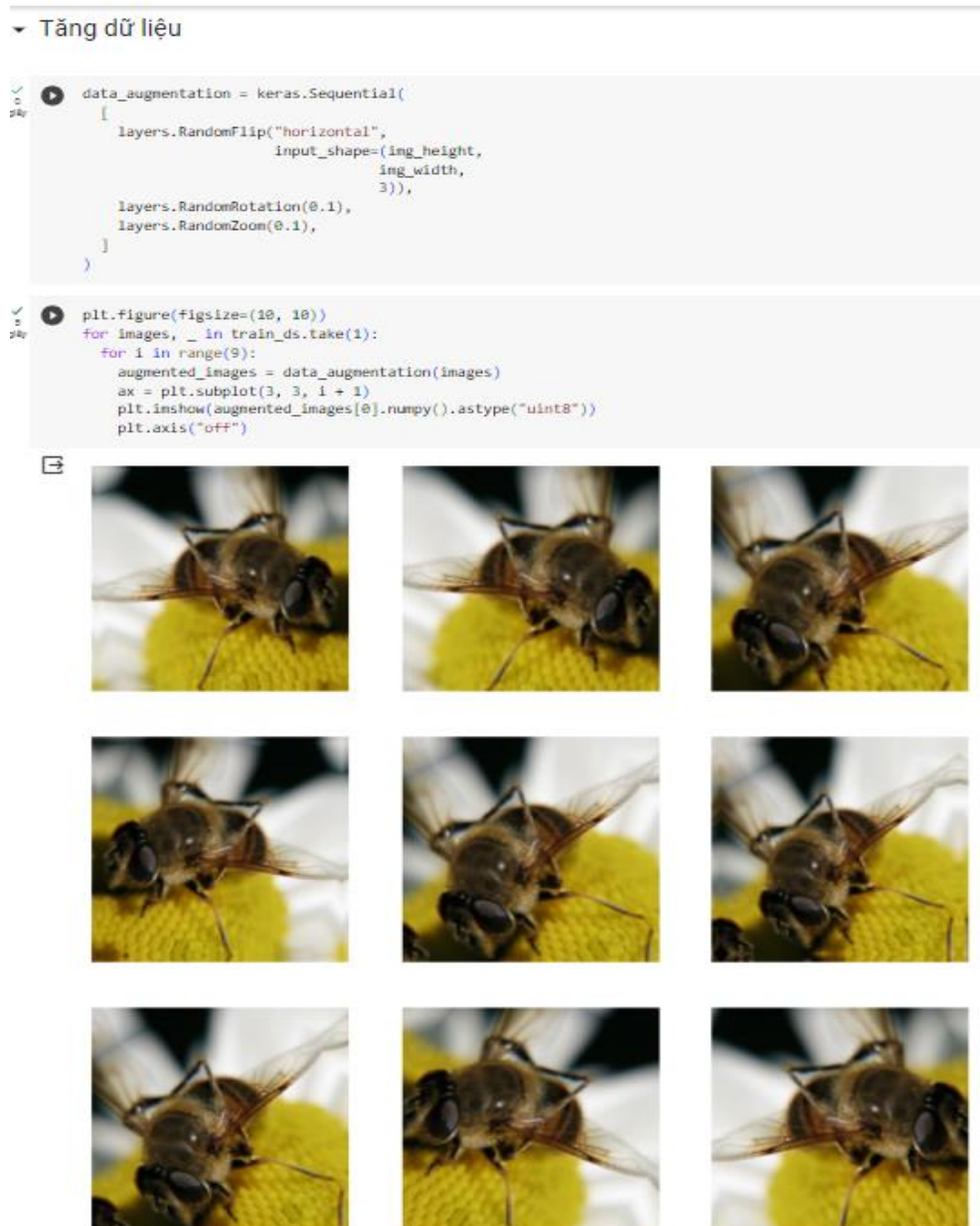
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower-right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper-right')
plt.title('Training and Validation Loss')
plt.show()
```



Hình 3.17: Kết quả đào tạo

Bước 12: Tăng dữ liệu: áp dụng phương pháp tạo dữ liệu đào tạo bổ sung từ các ví dụ hiện có của bạn bằng cách tăng cường chúng bằng cách sử dụng các phép biến đổi ngẫu nhiên mang lại hình ảnh trông đáng tin cậy:



Hình 3.18: Ảnh chụp nhiều góc độ sau khi làm tăng dữ liệu

Bước 13: Lọc bỏ: Khi bạn áp dụng dropout cho một lớp, nó sẽ ngẫu nhiên rơi ra (bằng cách đặt kích hoạt thành 0) một số đơn vị đầu ra từ lớp trong quá trình đào tạo. Bộ học lấy một số phân số làm giá trị đầu vào của nó, ở dạng chẳng hạn như 0,1, 0,2, 0,4, v.v. Điều này có nghĩa là bỏ 10%, 20% hoặc 40% đơn vị đầu ra ngẫu nhiên khỏi lớp được áp dụng.

```

model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])

```

Hình 3.19: Đoạn code lọc bỏ dữ liệu

Bước 14: Biên dịch lại mô hình sau khi lọc bỏ dữ liệu:

Biên dịch và đào tạo mô hình

```

[25] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

```

```

[26] model.summary()

```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential_1 (Sequential)	(None, 180, 180, 3)	0
rescaling_2 (Rescaling)	(None, 180, 180, 3)	0
conv2d_3 (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d_3 (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_4 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_4 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_5 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
flatten_1 (Flatten)	(None, 30976)	0
dense_2 (Dense)	(None, 128)	3965056
dense_3 (Dense)	(None, 5)	645

=====
 Total params: 3989285 (15.22 MB)
 Trainable params: 3989285 (15.22 MB)
 Non-trainable params: 0 (0.00 Byte)

Bước 14: Đào tạo mô hình lại:

```
[27] epochs = 15
     history = model.fit(
         train_ds,
         validation_data=val_ds,
         epochs=epochs
     )
```

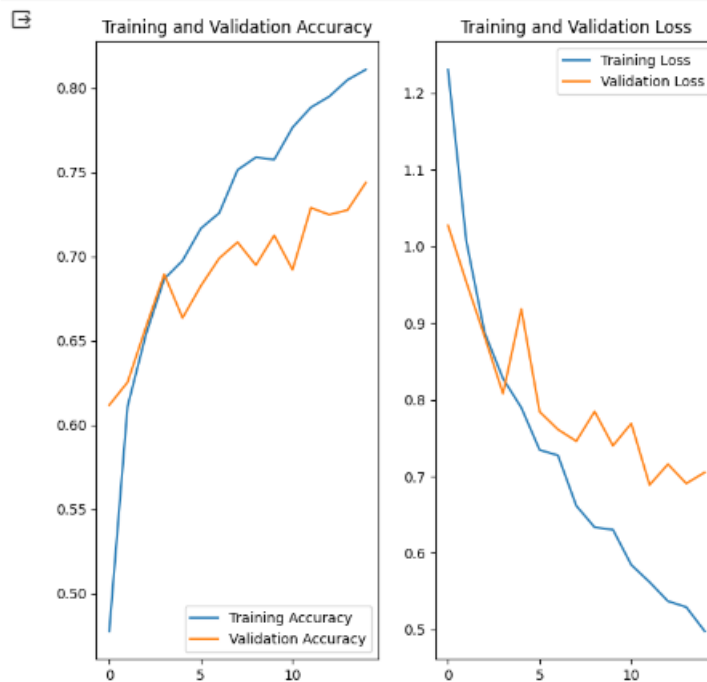
Epoch 1/15
92/92 [=====] - 6s 31ms/step - loss: 1.2309 - accuracy: 0.4775 - val_loss: 1.0275 - val_accuracy: 0.6117
Epoch 2/15
92/92 [=====] - 3s 31ms/step - loss: 1.0071 - accuracy: 0.6110 - val_loss: 0.9536 - val_accuracy: 0.6253
Epoch 3/15
92/92 [=====] - 3s 31ms/step - loss: 0.8877 - accuracy: 0.6536 - val_loss: 0.8824 - val_accuracy: 0.6580
Epoch 4/15
92/92 [=====] - 3s 30ms/step - loss: 0.8277 - accuracy: 0.6866 - val_loss: 0.8081 - val_accuracy: 0.6894
Epoch 5/15
92/92 [=====] - 3s 30ms/step - loss: 0.7898 - accuracy: 0.6975 - val_loss: 0.9183 - val_accuracy: 0.6635
Epoch 6/15
92/92 [=====] - 3s 29ms/step - loss: 0.7346 - accuracy: 0.7166 - val_loss: 0.7844 - val_accuracy: 0.6826
Epoch 7/15
92/92 [=====] - 3s 32ms/step - loss: 0.7275 - accuracy: 0.7258 - val_loss: 0.7613 - val_accuracy: 0.6989
Epoch 8/15
92/92 [=====] - 3s 30ms/step - loss: 0.6618 - accuracy: 0.7514 - val_loss: 0.7460 - val_accuracy: 0.7084
Epoch 9/15
92/92 [=====] - 3s 30ms/step - loss: 0.6336 - accuracy: 0.7589 - val_loss: 0.7847 - val_accuracy: 0.6948
Epoch 10/15
92/92 [=====] - 3s 33ms/step - loss: 0.6305 - accuracy: 0.7575 - val_loss: 0.7401 - val_accuracy: 0.7125
Epoch 11/15
92/92 [=====] - 3s 30ms/step - loss: 0.5846 - accuracy: 0.7766 - val_loss: 0.7691 - val_accuracy: 0.6921
Epoch 12/15
92/92 [=====] - 3s 29ms/step - loss: 0.5622 - accuracy: 0.7885 - val_loss: 0.6889 - val_accuracy: 0.7289
Epoch 13/15
92/92 [=====] - 3s 29ms/step - loss: 0.5373 - accuracy: 0.7950 - val_loss: 0.7160 - val_accuracy: 0.7248
Epoch 14/15
92/92 [=====] - 3s 33ms/step - loss: 0.5296 - accuracy: 0.8048 - val_loss: 0.6908 - val_accuracy: 0.7275
Epoch 15/15
92/92 [=====] - 3s 30ms/step - loss: 0.4982 - accuracy: 0.8110 - val_loss: 0.7052 - val_accuracy: 0.7439

Hình 3.20: Kết quả đào tạo lại mô hình sau khi lọc bỏ dữ liệu

Bước 15: Hình dung kết quả đào tạo:

▼ Hình dung kết quả đào tạo

```
✓ 1  
file acc = history.history['accuracy']  
val_acc = history.history['val_accuracy']  
  
loss = history.history['loss']  
val_loss = history.history['val_loss']  
  
epochs_range = range(epochs)  
  
plt.figure(figsize=(8, 8))  
plt.subplot(1, 2, 1)  
plt.plot(epochs_range, acc, label='Training Accuracy')  
plt.plot(epochs_range, val_acc, label='Validation Accuracy')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
  
plt.subplot(1, 2, 2)  
plt.plot(epochs_range, loss, label='Training Loss')  
plt.plot(epochs_range, val_loss, label='Validation Loss')  
plt.legend(loc='upper right')  
plt.title('Training and Validation Loss')  
plt.show()
```



Hình 3.21: Kết quả đào tạo

Bước 16: Dự đoán về bộ dữ liệu mới:

```


sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/592px-Red_sunflower.jpg"
sunflower_path = tf.keras.utils.get_file('Red_sunflower', origin=sunflower_url)

img = tf.keras.utils.load_img(
    sunflower_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)

```

 Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/592px-Red_sunflower.jpg
 117948/117948 [=====] - 0s 4us/step
 1/1 [=====] - 0s 197ms/step
 This image most likely belongs to sunflowers with a 99.59 percent confidence.

Hình 3.22: Kết quả đạt được độ tin cậy chính xác của đối tượng

Bước 17: Thực hiện phân loại 2 loài hoa daisy và tulip:

```

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Đường dẫn tới hai hình ảnh của bạn
image_path_1 = '/content/10791227_7168491604.jpg'
image_path_2 = '/content/5547758_eea9edfd54_n.jpg'

# Đọc hình ảnh từ đường dẫn
img1 = mpimg.imread(image_path_1)
img2 = mpimg.imread(image_path_2)

# Tạo subplot để hiển thị hai hình ảnh
plt.figure(figsize=(10, 5))

# Hiển thị hình ảnh thứ nhất
plt.subplot(1, 2, 1)
plt.imshow(img1)
plt.title('Image 1: loài hoa daisy')
plt.axis('off') # Tắt trục

# Hiển thị hình ảnh thứ hai
plt.subplot(1, 2, 2)
plt.imshow(img2)
plt.title('Image 2: loài hoa tulips')
plt.axis('off') # Tắt trục

plt.tight_layout() # Đảm bảo bố cục hợp lý
plt.show()

```

Image 1: loài hoa daisy

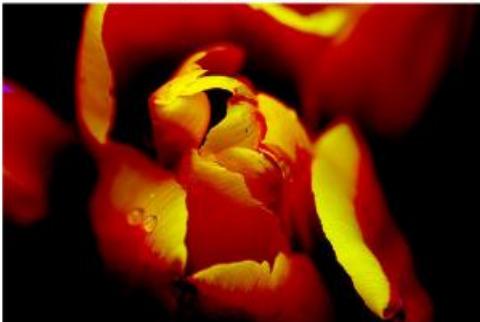

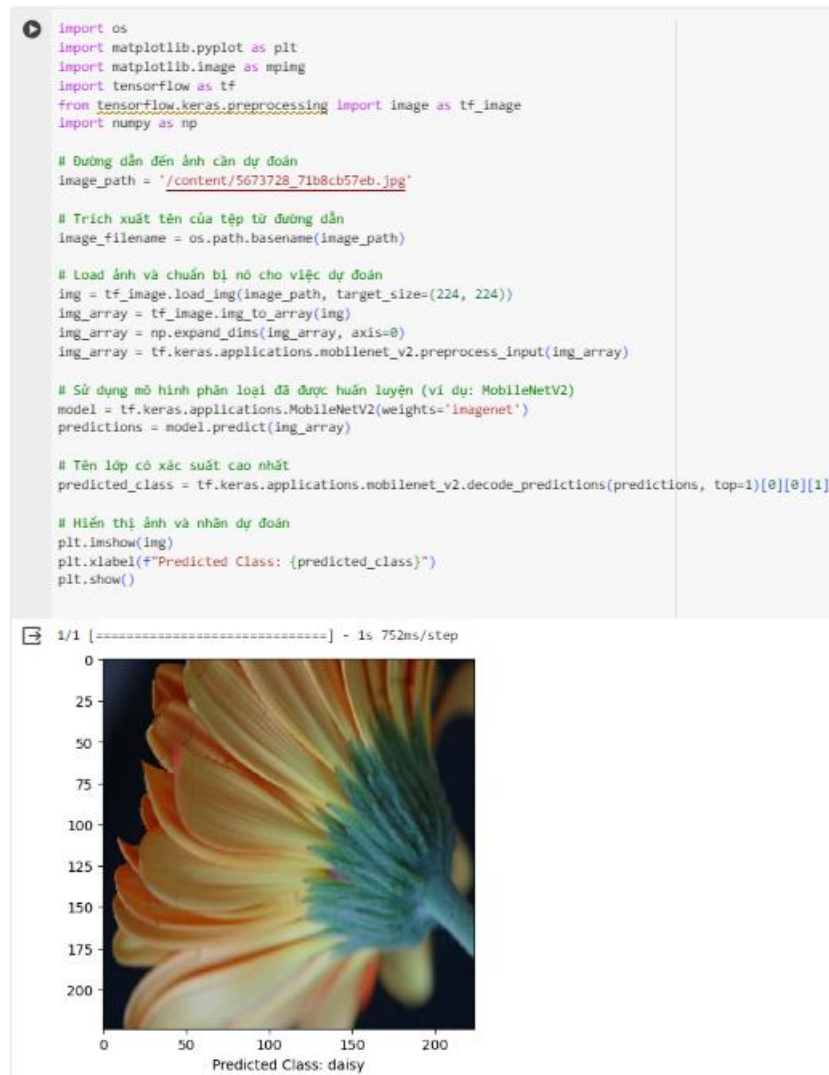


Image 2: loài hoa tulips



Hình 3.23: Phân loại 2 loài hoa daisy và tulips

Bước 18: Thực hiện load ảnh và in ra hình ảnh đã load:



Hình 3.24: Thực hiện load ảnh và đọc file ảnh đã load

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

Kết quả đạt được:

Sau khi nghiên cứu và thực hiện đề tài, đã hoàn thành được chương trình về đề tài “Phân loại ảnh dựa trên kỹ thuật học sâu”. Nó có thể giúp hình dung các mô hình của mạng neural trong phân loại đối tượng đặc trưng của ảnh.

Chương trình này sẽ là nền tảng cho tôi có thể áp dụng và phát triển dự án phần mềm với giao diện trực quan trong tương lai.

Hạn chế:

Với hệ thống này chỉ phục vụ chính cho việc phân loại những đặc trưng của đối tượng trong ảnh, nên khó khăn trong xử lý về mô hình phân loại các đặc trưng của đối tượng trong ảnh.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong tương lai, tôi định hướng sẽ phát triển đề tài Xây dựng chương trình " Phân loại ảnh dựa trên kỹ thuật học sâu " từ chương trình đã hoàn thành trước đó, trở thành hệ thống chạy trên hệ điều hành Window. Mục đích cuối cùng của tôi, hoàn thành cơ sở dữ liệu để giúp cho việc phân loại đối tượng trong ảnh có thể dễ dàng hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1]: Ths. Nguyễn Ngọc Giang, Đường vào lập trình python, Đại học quốc gia Hà Nội, Hà Nội
- [2]: <https://itnavi.com.vn/>
- [3]: <https://viblo.asia/p/mang-neural-network-WAyK84zpKxX>
- [4]: <https://intech.vietnamworks.com/article/top-5-sach-python-khong-the-bo-qua-nam-2023>
- [5]: <https://www.w3schools.com/python>
- [6]: <https://www.python.org/>

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN
ĐỀ CƯƠNG CHI TIẾT
THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH

Tên đề tài: Phân loại ảnh dựa trên kỹ thuật học sâu

Giáo viên hướng dẫn: Nguyễn Mộng Hiền

Thời gian thực hiện: 13/11/2023 đến 31/12/2023

Sinh viên thực hiện: Võ Anh Duy 110120167

1. Nội dung đề tài:

Đề tài phân loại ảnh dựa trên kỹ thuật học sâu gồm các nội dung sau:

- 1) Tìm hiểu về phân loại ảnh,
- 2) Tìm hiểu mạng neural network trong python,
- 3) Phân tích về các thư viện của neural network trong phân loại ảnh,
- 4) Cài đặt các thư viện trong python,
- 5) Chuẩn bị dữ liệu mẫu thử, nhập dữ liệu thử nghiệm.

2. Đối tượng nghiên cứu: Ảnh của một đối tượng cụ thể

3. Phương pháp nghiên cứu:

3.1. Nghiên cứu lý thuyết:

- Tổng quan neural network, deep learning
- Tìm hiểu cách thức hoạt động của neural network
- Các hàm dùng phân loại ảnh trong python.

3.2. Nghiên cứu thực nghiệm:

- Chuẩn bị dữ liệu ảnh
- Xây dựng mô hình
- Kiểm tra và triển khai

4. Phạm vi nghiên cứu: phân loại ảnh với tập dữ liệu thử nghiệm của một đối tượng dựa vào neural network

5. Phương pháp thực hiện:

- Nghiên cứu tài liệu về: Phân loại ảnh dựa trên kỹ thuật học sâu, công cụ thực hiện: Python.
- Thực nghiệm: cài đặt và kiểm thử dữ liệu.

6. Yêu cầu:

Hệ thống thực hiện các nội dung của đề tài gồm:

6.1. Thực hiện phân loại ảnh trên hệ thống:

- a. Thực hiện phân loại ảnh trên hệ thống:
 - 1) Thực hiện mô hình neural network trong deep learning theo tên ảnh của một loại ảnh cụ thể. Hệ thống hỗ trợ phân loại ảnh của một đối tượng theo neural network.
 - 2) Xuất ra hình ảnh của một đối tượng.

7. Kết quả đạt được: Hoàn thành các nội dung của đề tài như phần nội dung đã trình bày: Phân loại ảnh dựa trên kỹ thuật học sâu.

Kế hoạch thực hiện:

Tuần	Thời gian thực hiện	Nội dung công việc	Người thực hiện
1	Từ ngày 13/11/2023 đến 19/11/2023	<ul style="list-style-type: none">- Nghiên cứu tài liệu về phân loại ảnh dựa trên kỹ thuật học sâu- Tìm hiểu về neural network- Tìm hiểu công cụ thực hiện phân loại ảnh bằng python	Võ Anh Duy

2	Từ ngày 27/11/2023 đến 03/12/2023	<div><div>- Tìm hiểu ngôn ngữ python và các thư viện của python trong phân loại ảnh</div><div>- Tìm hiểu về mô hình trong phân loại ảnh</div><div>- Tìm hiểu về cách thức hoạt động của mô hình trong phân loại ảnh</div></div>	Võ Anh Duy
3	Từ ngày 11/12/2023 đến 17/12/2023	Viết chương trình phân loại ảnh	Võ Anh Duy
4	Từ ngày 25/12/2023 đến 31/12/2023	Hoàn thiện bài báo cáo	Võ Anh Duy

Xác nhận của GVHD

Ngày 05 tháng 11 năm 2023

Sinh viên thực hiện

Nguyễn Mộng Hiền

Võ Anh Duy