**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

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

**Đề tài số 15: Xây dựng hệ thống phát triển mô hình phân loại ảnh để nhận diện các loài động vật**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | **20210844** | **Đinh Văn Duy** | **DCCNTT12.10.3** |
| **2** | **20210809** | **Đinh Văn Ngân** | **DCCNTT12.10.3** |
| **3** | **20210851** | **Nguyễn Quang Thắng** | **DCCNTT12.10.3** |

**Bắc Ninh, năm 2024**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÀI TẬP LỚN**

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

**Đề tài số 15: Xây dựng hệ thống phát triển mô hình phân loại ảnh để nhận diện các loài động vật**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | **20210844** | **Đinh Văn Duy** | **DCCNTT12.10.3** |
| **2** | **20210809** | **Đinh Văn Ngân** | **DCCNTT12.10.3** |
| **3** | **20210851** | **Nguyễn Quang Thắng** | **DCCNTT12.10.3** |

|  |  |
| --- | --- |
|  |  |

**Bắc Ninh, năm 2024**

|  |  |
| --- | --- |
| **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**  **KHOA CÔNG NGHỆ THÔNG TIN** | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ 1, NĂM HỌC 2024** – **2025** |

|  |  |
| --- | --- |
| **PHIẾU CHẤM THI BÀI TẬP LỚN KẾT THÚC HỌC PHẦN**  **Mã đề thi:15**  **Tên học phần:**  **Xử lý ảnh và thị giác máy tính**  **Lớp Tín chỉ:**  **XATGMT.03.K12.03.LH.C04.1\_LT** | |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ tên)*  **Lương Thị Hồng Lan** | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ tên)* |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **Đinh Văn Duy** | **Đinh văn Ngân** | **Nguyễn Quang Thắng** |
| --- | --- | --- | --- | --- | --- |
| **20210844** | **20210809** | **20210851** |
| **1** | **Nội dung báo cáo trên Word đầy đủ** | **3.5** |  |  |  |
| 1.1 | Có bố cục rõ ràng (mục lục, phần mở đầu, nội dung chính, kết luận). | 0,5 |  |  |  |
| 1.2 | Nội dung phân tích rõ ràng, logic. | 0,5 |  |  |  |
| 1.3 | Có dẫn chứng, số liệu minh họa đầy đủ. | 0,5 |  |  |  |
| 1.4 | Ngôn ngữ và trình bày chuẩn, không lỗi chính tả. | 0,5 |  |  |  |
| 1.5 | Có trích dẫn tài liệu tham khảo đúng quy cách. | 0,5 |  |  |  |
| 1.6 | Được trình bày chuyên nghiệp (canh lề, font chữ, khoảng cách dòng hợp lý). | 0,5 |  |  |  |
| 1.7 | Tài liệu đầy đủ, bám sát yêu cầu của đề bài. | 0,5 |  |  |  |
| **2** | **Nội dung thuyết trình đầy đủ** | **1.0** |  |  |  |
| 2.1 | Trình bày tự tin, phát âm rõ ràng, mạch lạc. | 0,5 |  |  |  |
| 2.2 | Nội dung thuyết trình đúng trọng tâm, không lan man. | 0,5 |  |  |  |
| **3** | **Slides báo cáo đầy đủ nội dung + Hỏi đáp** | **3.0** |  |  |  |
| 3.1 | Slides có bố cục rõ ràng (mở đầu, nội dung, kết luận). | 0,5 |  |  |  |
| 3.2 | Thiết kế slides đẹp, chuyên nghiệp (màu sắc, hình ảnh minh họa). | 0,5 |  |  |  |
| 3.3 | Nội dung trên slides ngắn gọn, dễ hiểu, súc tích. | 0,5 |  |  |  |
| 3.4 | Nội dung slides phù hợp với nội dung báo cáo. | 0,5 |  |  |  |
| 3.5 | Trả lời câu hỏi đầy đủ, chính xác. | 0,5 |  |  |  |
| 3.6 | Trả lời câu hỏi tự tin, thuyết phục. | 0,5 |  |  |  |
| **4** | **Code đầy đủ** | **2.5** |  |  |  |
| 1.1 | Code được trình bày rõ ràng, có chú thích đầy đủ. | 0,5 |  |  |  |
| 1.2 | Code chạy đúng, không lỗi. | 0,5 |  |  |  |
| 1.3 | Code tối ưu, không dư thừa. | 0,5 |  |  |  |
| 1.4 | Đáp ứng đầy đủ các yêu cầu chức năng theo đề bài. | 0,5 |  |  |  |
| 1.5 | Có tính sáng tạo hoặc cải thiện so với yêu cầu. | 0,5 |  |  |  |
| **TỔNG ĐIỂM BẰNG SỐ:** | | **10** |  |  |  |
| **TỔNG ĐIỂM BẰNG CHỮ:** | | *Mười tròn* |  |  |  |

MỤC LỤC

[CHƯƠNG 1 CÁC KIẾN THỨC CƠ SỞ 8](#_Toc184716452)

[**1.1 Nhận dạng đối tượng 8**](#_Toc184716453)

[**1.2 Các kỹ thuật sử dụng trong bài toán nhận dạng 10**](#_Toc184716454)

[**1.2.1 Học Có Giám sát (Supervised Learning) 10**](#_Toc184716455)

[**1.2.2 Học Không Giám sát (Unsupervised Learning) 11**](#_Toc184716456)

[**1.2.3 Học Tăng Cường (Reinforcement Learning) 12**](#_Toc184716457)

[**1.2.4 Học sâu (Deep Learning) 13**](#_Toc184716458)

[**1.3 Ngôn ngữ lập trình và các thư viện sử dụng 14**](#_Toc184716459)

[**1.3.1 Python 14**](#_Toc184716460)

[**1.3.2 Các thư viện 15**](#_Toc184716461)

[CHƯƠNG 2. XÂY DỰNG HỆ THỐNG 18](#_Toc184716462)

[**2.1 Mô tả bài toán 18**](#_Toc184716463)

[**2.2 Mô hình MobileNetV2 18**](#_Toc184716464)

[**2.2.1 Depthwise Separable Convolution 19**](#_Toc184716465)

[**2.2.2. Linear Bottleneck 19**](#_Toc184716466)

[**2.2.3. Inverted Residual Block 20**](#_Toc184716467)

[**2.2.4. Batch Normalization (BN) 20**](#_Toc184716468)

[**2.2.5. ReLU6 Activation Function 21**](#_Toc184716469)

[**2.2.6. Lightweight Design (Thiết kế gọn nhẹ) 21**](#_Toc184716470)

[**2.2.7 Thu thập dữ liệu: Tập hợp hình ảnh của các loài động vật cần phân loại. 21**](#_Toc184716471)

[**2.2.8 Tiền xử lý dữ liệu: 21**](#_Toc184716472)

[**2.3 Quy trình xử lý 22**](#_Toc184716473)

[**2.4 Xây dựng mô hình và huần luyện 23**](#_Toc184716474)

[**2.5 Thiết kế giao diện 26**](#_Toc184716475)

[**2.6 Cấu trúc chương trình 29**](#_Toc184716476)

[CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT QUẢ 38](#_Toc184716477)

[**3.1 Bộ dữ liệu 38**](#_Toc184716478)

[**3.2 Mô hình sử dụng 42**](#_Toc184716479)

[**3.3 Kết quả dự đoán 42**](#_Toc184716480)

[**3.4 Đánh giá hiệu suất 47**](#_Toc184716481)

[KẾT LUẬN 48](#_Toc184716482)

[TÀI LIỆU THAM KHẢO 49](#_Toc184716483)

DANH SÁCH HÌNH ẢNH

|  |  |  |
| --- | --- | --- |
| STT | Tên hình ảnh | Số trang |
| 1 | Hình 1.1 1 Các giai đoạn cho bài toán nhận dạng | 8 |
| 2 | Hình 1.2 1 Mô phỏng kỹ thuật học có giám sát | 10 |
| 3 | Hình 1.2 2 Mô phỏng kỹ thuật học không giám sát | 11 |
| 4 | Hình 1.2 3 Mô phỏng kỹ thuật học tăng cường | 12 |
| 5 | Hình 1.2 4 Mô phỏng kỹ thuật học sâu | 13 |
| 6 | Hình 1.1. Giao diện chưa phân loại động vật | 25 |
| 7 | Hình 1.2. Giao diện đã phân loại động vật | 27 |
| 8 | Hình 1.3. Ảnh dự đoán test lần 1 | 39 |
| 9 | Hình 1.4. Ảnh dự đoán test lần 2 | 39 |
| 10 | Hình 1.5. Ảnh dự đoán test lần 3 | 40 |
| 11 | Hình 1.6. Ảnh dự đoán test lần 4 | 40 |

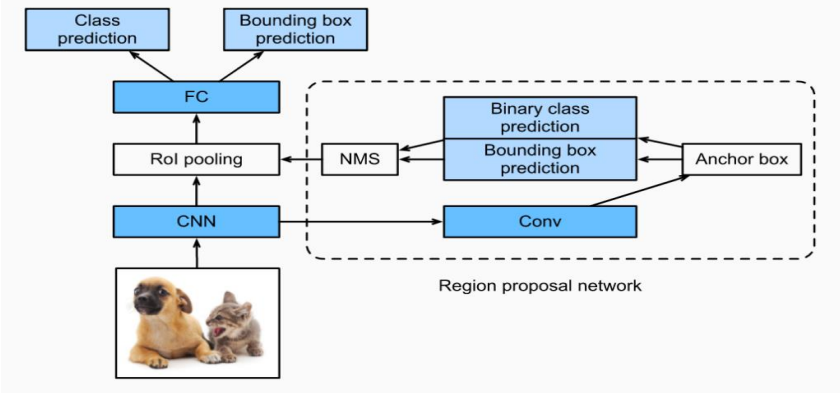
DANH SÁCH BẢNG BIỂU

|  |  |  |
| --- | --- | --- |
| stt | hình ảnh | số trang |
| 1 | Bảng dự đoán test lần 1 | 42 |
| 2 | Bảng dự đoán test lần 2 | 42 |
| 3 | Bảng dự đoán test lần 3 | 43 |
| 4 | Bảng dự đoán test lần 4 | 43 |

CHƯƠNG 1 CÁC KIẾN THỨC CƠ SỞ

* 1. **Nhận dạng đối tượng**

Bài toán nhận dạng là một lĩnh vực rộng lớn trong trí tuệ nhân tạo, bao gồm việc máy

tính tự động phân loại, phân nhóm hoặc xác định các đối tượng, sự kiện, hoặc các mẫu trong dữ liệu. Để hoàn thành một bài toán nhận dạng, chúng ta thường trải qua các bước sau:

Hình 1.1 1 Các giai đoạn cho bài toán nhận dạng

Bước 1. Thu thập và chuẩn bị dữ liệu

Thu thập: Tìm kiếm và thu thập một lượng lớn dữ liệu đại diện cho các đối tượng cần nhận dạng. Ví dụ: nếu muốn nhận dạng con vật, cần thu thập nhiều hình ảnh con vật khác nhau.

Làm sạch: Loại bỏ dữ liệu nhiễu, không chính xác hoặc trùng lặp.

Đánh nhãn: Gán nhãn cho từng dữ liệu để máy tính hiểu được nó thuộc loại nào. Ví dụ: trong nhận dạng động vật, mỗi hình ảnh sẽ được gán nhãn là "con A", "con B",...

Chia dữ liệu: Chia dữ liệu thành ba tập: tập huấn luyện (để huấn luyện mô hình), tập kiểm tra (để đánh giá mô hình trong quá trình huấn luyện), và tập kiểm định (để đánh giá mô hình cuối cùng).

Bước 2. Trích xuất đặc trưng

Chọn các đặc trưng: Lựa chọn những đặc trưng quan trọng nhất để phân biệt các đối tượng. Ví dụ: trong nhận dạng hình ảnh, các đặc trưng có thể là màu sắc, hình dạng… Trích xuất: Sử dụng các thuật toán để trích xuất các đặc trưng từ dữ liệu thô.

Bước 3. Chọn mô hình Lựa chọn:

Chọn một mô hình phù hợp với bài toán. Các mô hình phổ biến bao gồm:

Mô hình dựa trên thống kê: Naive Bayes, SVM

Mạng thần kinh nhân tạo: CNN (Convolutional Neural Network), RNN (Recurrent Neural Network)

Mô hình dựa trên cây quyết định: Decision Tree, Random Forest

Huấn luyện: Dùng tập dữ liệu huấn luyện để huấn luyện mô hình. Trong quá trình này, mô hình sẽ học cách liên kết giữa các đặc trưng và nhãn.

Bước 4. Đánh giá mô hình:

Sử dụng tập kiểm tra: Đánh giá độ chính xác của mô hình trên tập kiểm tra.

Tính các chỉ số: Tính các chỉ số đánh giá như độ chính xác, độ nhạy, độ đặc hiệu,... để đánh giá hiệu suất của mô hình.

Bước 5. Điều chỉnh và tối ưu:

Điều chỉnh siêu tham số: Thay đổi các siêu tham số của mô hình để cải thiện hiệu suất. Thay đổi mô hình: Nếu cần, có thể thay đổi mô hình hoặc phương pháp trích xuất đặc trưng.

Bước 6. Triển khai:

Tích hợp: Tích hợp mô hình vào ứng dụng thực tế.

Dự đoán: Sử dụng mô hình để dự đoán nhãn của dữ liệu mới. Ví dụ: Nhận dạng hình ảnh

* 1. **Các kỹ thuật sử dụng trong bài toán nhận dạng**
     1. **Học Có Giám sát (Supervised Learning)**

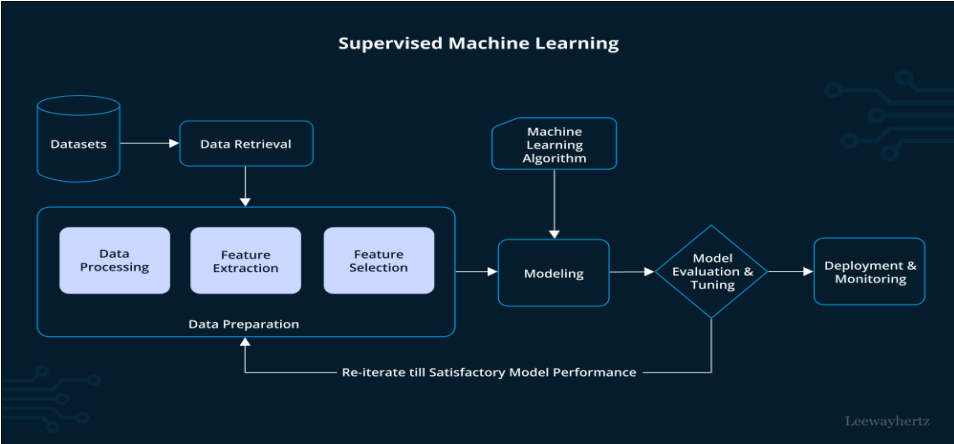
Học có giám sát (Supervised Learning) là một nhánh quan trọng trong học máy, nơi mô hình học từ dữ liệu đã được gắn nhãn trước. Nghĩa là, mỗi dữ liệu đầu vào sẽ đi kèm với một nhãn tương ứng, cho biết dữ liệu đó thuộc lớp nào.

Dữ liệu Đã Được Gắn Nhãn: Dữ liệu huấn luyện bao gồm các cặp (đầu vào, nhãn). Ví dụ, trong bài toán nhận dạng ảnh, đầu vào là một hình ảnh và nhãn là tên của đối tượng trong hình (mèo, chó, xe hơi).

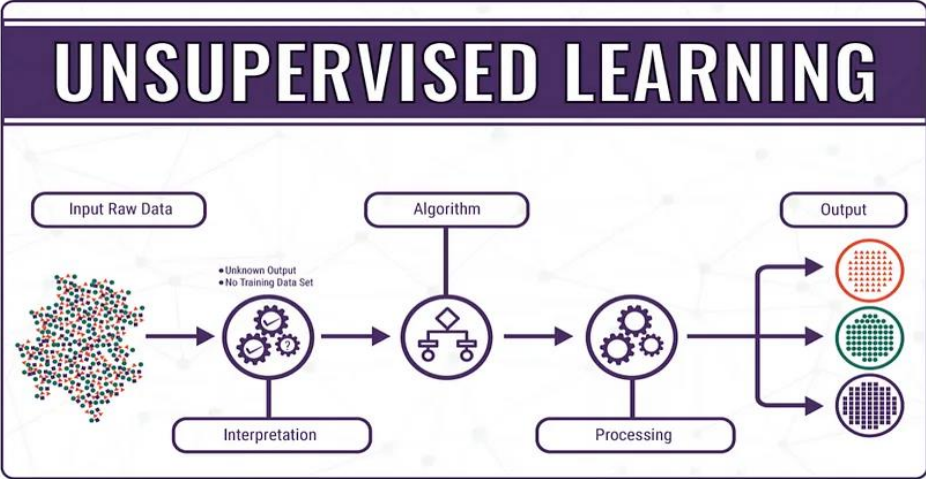
Học Mối Liên Hệ: Mô hình học để tìm ra mối liên hệ giữa đầu vào và nhãn. Mối liên hệ này được biểu diễn dưới dạng một hàm toán học.

Dự Đoán: Sau khi được huấn luyện, mô hình có thể dự đoán nhãn của một dữ liệu đầu vào mới chưa từng gặp.

Hình mô tả kỹ thuật

Hình 1.2 1 Mô phỏng kỹ thuật học có giám sát 

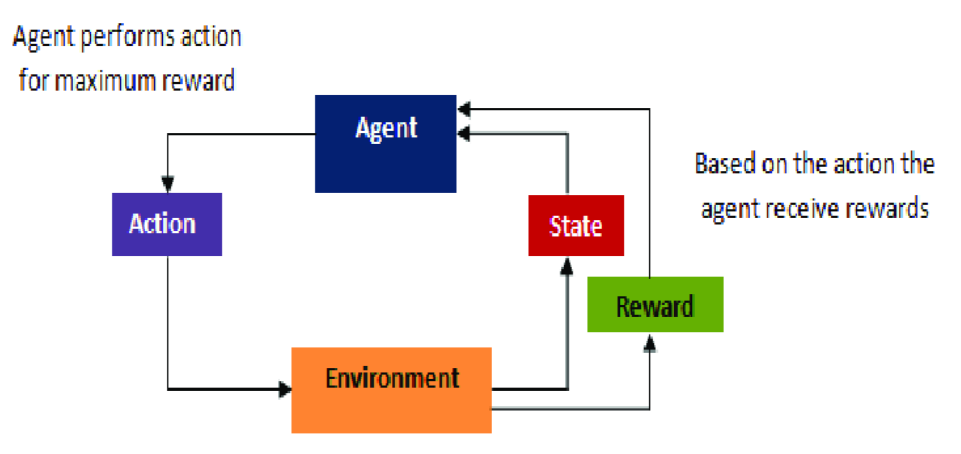
* + 1. **Học Không Giám sát (Unsupervised Learning)**

Học không giám sát (Unsupervised Learning) là một nhánh của học máy nơi mô hình học từ dữ liệu không được gắn nhãn. Khác với học có giám sát, ở đây chúng ta không biết trước các nhãn tương ứng với mỗi dữ liệu. Thay vào đó, mô hình tự tìm kiếm các cấu trúc ẩn, các mẫu trong dữ liệu để đưa ra những kết luận nhất định. Tư tưởng cơ bản Dữ liệu không gắn nhãn: Dữ liệu đầu vào không có nhãn tương ứng. Tìm kiếm cấu trúc ẩn: Mô hình tự tìm kiếm các nhóm, các cụm hoặc các quy tắc ẩn trong dữ liệu. Không có đáp án đúng: Không có một đáp án đúng duy nhất cho một bài toán học không giám sát, kết quả có thể phụ thuộc vào thuật toán và các tham số được chọn. Hình mô tả kỹ thuật

Hình 1.2 2 Mô phỏng kỹ thuật học không giám sát

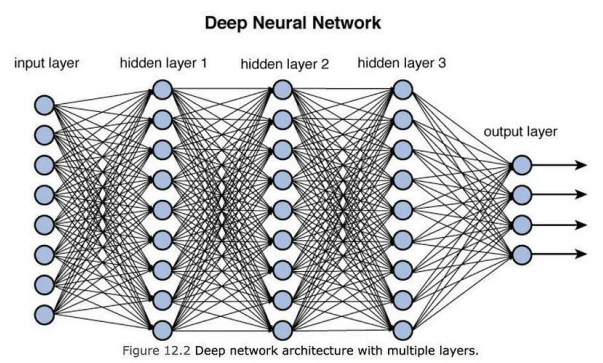
* + 1. **Học Tăng Cường (Reinforcement Learning)**

Học tăng cường (Reinforcement Learning - RL) là một nhánh của học máy, mô phỏng quá trình học hỏi của con người thông qua thử và sai. Trong RL, một tác nhân (agent) sẽ tương tác với một môi trường (environment) và học cách thực hiện các hành động để tối đa hóa phần thưởng (reward) nhận được. Tư tưởng cơ bản Tác nhân (Agent): Đây là thực thể đưa ra quyết định, ví dụ: một robot, một chương trình máy tính chơi game. Môi trường (Environment): Là thế giới mà tác nhân tương tác, có thể là một trò chơi, một robot mô phỏng, hoặc một hệ thống thực tế. Hành động (Action): Các lựa chọn mà tác nhân có thể thực hiện trong một trạng thái nhất định. Trạng thái (State): Mô tả tình huống hiện tại của môi trường. Phần thưởng (Reward): Một tín hiệu số cho biết hành động của tác nhân là tốt hay xấu. Mục tiêu của tác nhân là tối đa hóa tổng phần thưởng nhận được trong dài hạn. Hình mô tả kỹ thuật

Hình 1.2 3 Mô phỏng kỹ thuật học tăng cường 

* + 1. **Học sâu (Deep Learning)**

Học sâu là một nhánh của học máy, sử dụng các mạng thần kinh nhân tạo nhiều lớp để học trực tiếp từ dữ liệu, thường là dữ liệu không có cấu trúc như hình ảnh, âm thanh và văn bản. Nó được lấy cảm hứng từ cách hoạt động của não người, nơi thông tin được xử lý qua nhiều lớp neuron. Tư tưởng cơ bản Mạng thần kinh nhân tạo: Là một mô hình tính toán mô phỏng các neuron trong não người. Các neuron này được kết nối với nhau tạo thành các lớp, và thông tin được truyền đi qua các lớp này. Học các đặc trưng: Thay vì người lập trình phải xác định thủ công các đặc trưng, mạng thần kinh học sâu tự động học các đặc trưng từ dữ liệu. Các lớp đầu tiên học các đặc trưng đơn giản, các lớp sau học các đặc trưng phức tạp hơn. Học bằng cách backpropagation: Quá trình huấn luyện mạng thần kinh bao gồm việc so sánh kết quả dự đoán của mạng với nhãn thực tế, tính toán lỗi và điều chỉnh các trọng số của mạng để giảm thiểu lỗi. Quá trình này được lặp đi lặp lại nhiều lần. Hình mô tả kỹ thuật



Hình 1.2 4 Mô phỏng kỹ thuật học sâu

* 1. **Ngôn ngữ lập trình và các thư viện sử dụng**
     1. **Python**

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển. Những lợi ích của Python bao gồm:

‐ Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh.

‐ Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.

‐ Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Nhờ đó, các nhà phát triển sẽ không cần phải viết mã từ đầu.

‐ Các nhà phát triển có thể dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.

‐ Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

* + 1. **Các thư viện**

**Thư viện:**

* TensorFlow/Keras

TensorFlow là một thư viện mã nguồn mở của Google dành cho học máy và học sâu. Nó cung cấp nền tảng mạnh mẽ để xây dựng và triển khai các mô hình học sâu trên nhiều loại thiết bị, từ máy tính cá nhân đến hệ thống đám mây.

Keras là một API cấp cao của TensorFlow, giúp xây dựng các mô hình học sâu một cách dễ dàng và trực quan. Keras cung cấp giao diện thân thiện với người dùng, hỗ trợ định nghĩa mô hình theo hai cách: tuần tự (Sequential) và kiến trúc đồ thị (Functional API).

* Lợi ích khi sử dụng TensorFlow/Keras trong dự án:
* Xử lý ảnh hiệu quả: TensorFlow cung cấp các lớp và hàm hỗ trợ tiền xử lý ảnh, như tf.image.resize, tf.image.random\_flip\_left\_right.
* Hỗ trợ GPU/TPU: Tăng tốc huấn luyện và suy luận mô hình.
* Cộng đồng lớn: Có tài liệu phong phú, ví dụ mẫu và diễn đàn hỗ trợ.
* Tích hợp mô hình có sẵn: Các mô hình học sâu phổ biến như MobileNetV2 có thể được sử dụng thông qua tf.keras.applications.
* Các tính năng chính được sử dụng trong dự án:
* Hỗ trợ xử lý ảnh: Chuẩn hóa dữ liệu và chia tập dữ liệu thành các batch.
* Xây dựng mô hình học sâu: Triển khai MobileNetV2 để phân loại ảnh.
* Fine-tuning: Tinh chỉnh mô hình MobileNetV2 trên tập dữ liệu động vật.
* NumPy

NumPy là thư viện mạnh mẽ dành cho tính toán khoa học trong Python, hỗ trợ các phép toán trên mảng (array) và ma trận (matrix) đa chiều. NumPy thường được sử dụng trong các ứng dụng xử lý dữ liệu, phân tích, và học máy.

* Lợi ích khi sử dụng NumPy trong dự án:
* Xử lý dữ liệu nhanh chóng: Tăng tốc các phép toán như cộng, trừ, nhân, chia trên ma trận ảnh.
* Tích hợp tốt với TensorFlow: NumPy và TensorFlow dễ dàng chuyển đổi dữ liệu qua lại.
* Dễ dàng thao tác với ảnh: NumPy hỗ trợ đọc, xử lý và chuẩn hóa các ma trận ảnh.
* Tính năng chính được sử dụng trong dự án:
* Biến đổi ảnh: Chuyển đổi ảnh từ định dạng PIL sang mảng NumPy để xử lý và chuẩn hóa.
* Phép toán ma trận: Thực hiện các phép toán cần thiết trong xử lý ảnh.
* PIL (Python Imaging Library)

Pillow (PIL) là thư viện xử lý ảnh mạnh mẽ trong Python, hỗ trợ các chức năng như mở, lưu, và chỉnh sửa ảnh. Đây là một công cụ không thể thiếu trong các dự án xử lý ảnh nhờ tính đơn giản và hiệu quả.

* Lợi ích khi sử dụng PIL trong dự án:
* Hỗ trợ định dạng ảnh đa dạng: Bao gồm JPEG, PNG, BMP, GIF,...
* Chỉnh sửa ảnh dễ dàng: Thay đổi kích thước, cắt, xoay và áp dụng các hiệu ứng.
* Tích hợp tốt với NumPy: Dễ dàng chuyển đổi giữa ảnh PIL và mảng NumPy.
* Tính năng chính được sử dụng trong dự án:
* Thay đổi kích thước ảnh: Đảm bảo tất cả ảnh đầu vào phù hợp với kích thước yêu cầu của mô hình.
* Chuyển đổi định dạng: Đọc và chuyển đổi ảnh sang dạng NumPy để xử lý.
* PyQt5

PyQt5 là thư viện phát triển giao diện người dùng (GUI) mạnh mẽ trong Python, cung cấp các công cụ tạo ứng dụng desktop với giao diện trực quan và chuyên nghiệp. PyQt5 được xây dựng dựa trên Qt Framework, cho phép tạo ra các giao diện hiện đại và đa nền tảng.

* Lợi ích khi sử dụng PyQt5 trong dự án:
* Giao diện thân thiện: Tạo GUI trực quan để người dùng tải và phân loại ảnh.
* Tích hợp với xử lý ảnh: PyQt5 dễ dàng kết hợp với các thư viện như TensorFlow và PIL để hiển thị kết quả phân loại.
* Hỗ trợ đa nền tảng: Ứng dụng phát triển với PyQt5 có thể chạy trên Windows, macOS, và Linux.
* Tính năng chính được sử dụng trong dự án:
* Hiển thị ảnh: Sử dụng các widget như QLabel, QPixmap để hiển thị ảnh đầu vào và kết quả phân loại.
* Tương tác người dùng: Sử dụng QPushButton và QFileDialog để cho phép người dùng chọn ảnh và thực hiện phân loại.
* Kết nối với TensorFlow: Gửi dữ liệu ảnh đến mô hình MobileNetV2 và hiển thị kết quả phân loại.

**Mô hình học sâu:**

* MobileNetV2: Là một mô hình học sâu nhẹ, được tối ưu hóa để hoạt động trên các thiết bị có tài nguyên hạn chế. Được sử dụng rộng rãi trong các bài toán phân loại ảnh nhờ tính hiệu quả và độ chính xác cao.

**Phần mềm và công cụ hỗ trợ:**

* PyCharm: IDE chính được sử dụng để phát triển và quản lý mã nguồn của ứng dụng.

# CHƯƠNG 2. XÂY DỰNG HỆ THỐNG

## 2.1 Mô tả bài toán

Bài toán phân loại động vật nhằm xây dựng một hệ thống trí tuệ nhân tạo có khả năng nhận diện và phân loại hình ảnh của các loài động vật khác nhau từ các hình ảnh đầu vào. Hệ thống này sẽ dự đoán loài động vật trong ảnh và cung cấp thông tin chi tiết như độ chính xác (confidence) cho từng dự đoán.

Đầu vào:

Hình ảnh của động vật, được cung cấp bởi người dùng (các định dạng phổ biến như .jpg, .png, .bmp).

Dữ liệu huấn luyện: Bộ dữ liệu ảnh của các loài động vật đã được gán nhãn trước, với các lớp cụ thể như:

Bird (Chim)

Cat (Mèo)

Chicken (Gà)

Dog (Chó)

Elephant (Voi)

Butterfly (Bướm)

Horse (Ngựa)

Spider (Nhện)

Turtle (Rùa)

Đầu ra:

Kết quả dự đoán:

Loài động vật trong ảnh.

Độ tự tin của mô hình cho từng dự đoán.

Danh sách các lớp được sắp xếp theo xác suất từ cao đến thấp.

## 2.2 Mô hình MobileNetV2

MobileNetV2 được xây dựng trên các ý tưởng chính sau đây, giúp tối ưu hóa hiệu quả tính toán và tài nguyên mà vẫn đảm bảo độ chính xác cao trong các bài toán thị giác máy tính. Dưới đây là phân tích chi tiết từng tư tưởng kỹ thuật cùng với ưu, nhược điểm.

### 2.2.1 Depthwise Separable Convolution

Depthwise Separable Convolution là một cải tiến so với tích chập truyền thống, chia quá trình tích chập thành hai bước:

Depthwise Convolution: Áp dụng một bộ lọc riêng lẻ trên mỗi kênh (channel) của đầu vào.

Pointwise Convolution: Áp dụng tích chập 1×11 \times 11×1 để kết hợp các kênh với nhau.

Ưu điểm:

Giảm số lượng tham số: Tích chập này giảm đáng kể số lượng tham số cần thiết (giảm khoảng 8–9 lần so với tích chập truyền thống).

Tăng tốc độ xử lý: Số lượng phép toán được giảm đi rất nhiều, phù hợp với các thiết bị di động và nhúng.

Nhược điểm:

Hạn chế trích xuất đặc trưng không gian: Depthwise Convolution chỉ hoạt động trên từng kênh riêng lẻ, làm giảm khả năng tổng hợp thông tin giữa các kênh.

Độ chính xác thấp hơn so với tích chập truyền thống khi số lượng dữ liệu lớn hoặc bài toán phức tạp.

### 2.2.2. Linear Bottleneck

Linear Bottleneck là lớp hẹp (bottleneck layer) được sử dụng để giảm chiều không gian của các đặc trưng. Không giống như các lớp tích chập thông thường (có hàm kích hoạt ReLU), Linear Bottleneck chỉ áp dụng phép tính tuyến tính để tránh mất thông tin.

Ưu điểm:

Tránh mất mát thông tin: Sử dụng phép tính tuyến tính thay vì ReLU, giúp giữnguyên các thông tin cần thiết, đặc biệt là ở các lớp đầu ra.

Giảm kích thước mạng: Linear Bottleneck giúp giảm số lượng tham số và kích thước tổng thể của mạng.

Nhược điểm:

Khả năng phi tuyến giảm: Việc loại bỏ ReLU có thể làm giảm khả năng biểu diễn phi tuyến của mô hình.

Hiệu quả thấp hơn khi dữ liệu phức tạp: Với các bài toán đòi hỏi khả năng trích xuất đặc trưng sâu hơn, Linear Bottleneck có thể không đủ mạnh.

### 2.2.3. Inverted Residual Block

Inverted Residual Block (Khối dư đảo ngược) là cải tiến dựa trên khối dư (residual block) trong ResNet, nhưng với cách tiếp cận ngược lại:

Đầu vào được mở rộng ra một không gian chiều cao hơn trước khi áp dụng tích chập.

Sau đó, kết quả được nén lại để giảm kích thước đầu ra.

Ưu điểm:

Tối ưu hóa hiệu suất: Việc mở rộng chiều trước khi tính toán giúp mô hình trích xuất được nhiều đặc trưng hơn.

Giảm số lượng tham số: Việc nén lại sau khi xử lý giúp giảm tài nguyên tính toán.

Tăng khả năng xử lý phi tuyến: Kết hợp tốt giữa các phép mở rộng, tích chập và nén, giúp duy trì độ chính xác cao.

Nhược điểm:

Phức tạp trong thiết kế: Việc mở rộng rồi nén có thể khó hiểu và cần tối ưu hóa cẩn thận.

Tăng kích thước tạm thời của tensor: Khi mở rộng không gian chiều, kích thước tensor trong quá trình xử lý tăng lên, tiêu tốn thêm bộ nhớ.

### 2.2.4. Batch Normalization (BN)

Batch Normalization được sử dụng để chuẩn hóa dữ liệu qua các lớp, giúp giảm hiện tượng gradient biến mất hoặc bùng nổ.

Ưu điểm:

Tăng độ ổn định trong huấn luyện: BN giúp mô hình hội tụ nhanh hơn.

Tăng hiệu suất tổng thể: BN làm giảm nhu cầu khởi tạo trọng số cẩn thận và cải thiện khả năng tổng quát hóa.

Nhược điểm:

Chi phí tính toán tăng: Thêm các phép toán chuẩn hóa làm tăng chi phí tính toán và thời gian xử lý.

Kém hiệu quả trên dữ liệu nhỏ: BN hoạt động kém hiệu quả khi kích thước batch quá nhỏ (batch size < 16).

### 2.2.5. ReLU6 Activation Function

MobileNetV2 sử dụng hàm kích hoạt ReLU6 thay vì ReLU thông thường. ReLU6 giới hạn giá trị đầu ra trong khoảng [0,6][0, 6][0,6], giúp mô hình ổn định hơn khi sử dụng trong các thiết bị di động (quantized networks).

Ưu điểm:

Ổn định hơn với các mô hình lượng tử hóa (Quantization): Giá trị trong khoảng [0,6][0, 6][0,6] phù hợp với các phép tính số nguyên trên phần cứng nhúng.

Giảm nguy cơ chết ReLU (Dead ReLU): Giới hạn đầu ra ngăn việc gradient bị triệt tiêu hoàn toàn.

Nhược điểm:

Giới hạn độ linh hoạt: ReLU6 hạn chế giá trị đầu ra, đôi khi làm mất đi khả năng biểu diễn của mô hình.

### 2.2.6. Lightweight Design (Thiết kế gọn nhẹ)

MobileNetV2 được thiết kế tối ưu với số lượng tham số và phép tính thấp, giúp nó phù hợp với các thiết bị có tài nguyên hạn chế.

Ưu điểm:

Chạy nhanh trên thiết bị di động: Với số lượng tham số nhỏ và FLOPs thấp, MobileNetV2 rất nhanh khi chạy trên CPU hoặc GPU của điện thoại.

Thân thiện với triển khai: Kích thước mô hình nhỏ gọn, dễ dàng triển khai trên các thiết bị nhúng.

Nhược điểm:

Hiệu suất giảm trên bài toán phức tạp: Với số lượng tham số ít, MobileNetV2 không đạt hiệu suất cao như các mô hình lớn như ResNet hoặc DenseNet.

Giảm độ chính xác: Trong một số trường hợp, tối ưu hóa cho tốc độ khiến mô hình hy sinh độ chính xác.

### 2.2.7 Thu thập dữ liệu: Tập hợp hình ảnh của các loài động vật cần phân loại.

### 2.2.8 Tiền xử lý dữ liệu:

Chuyển đổi kích thước hình ảnh về một kích thước cố định (ví dụ: 224x224 pixels).

Chuẩn hóa pixel (scale các giá trị về khoảng [0, 1]).

Tăng cường dữ liệu (data augmentation) để cải thiện khả năng tổng quát hóa của mô hình, ví dụ: xoay ảnh, lật ngang, phóng to/thu nhỏ.

Xây dựng mô hình:

Mô hình sử dụng: MobileNetV2 (hoặc các mạng CNN khác) được chọn vì tính hiệu quả và khả năng xử lý tốt với tài nguyên hạn chế.

Cấu trúc mô hình:

Sử dụng mô hình MobileNetV2 với trọng số imagenet làm lớp nền (base model).

Thêm các lớp tùy chỉnh (fully connected layers) phía trên để dự đoán 9 lớp động vật.

Sử dụng hàm kích hoạt softmax ở lớp đầu ra để đưa ra xác suất cho từng lớp.

Huấn luyện mô hình:

Hàm mất mát: categorical\_crossentropy cho bài toán phân loại đa lớp.

Bộ tối ưu hóa: Adam để tăng tốc quá trình hội tụ.

Callbacks:

EarlyStopping: Dừng huấn luyện sớm nếu mô hình không cải thiện.

ReduceLROnPlateau: Giảm tốc độ học nếu mô hình ngừng cải thiện.

## 2.3 Quy trình xử lý

* **Chuẩn bị dữ liệu**

Mục tiêu:

Chuẩn bị dữ liệu đầu vào cho mô hình để phân loại ảnh động vật.

Phân tích:

Đường dẫn dữ liệu: Dữ liệu ảnh được lưu tại data\_dir (D:\PhanLoaiDongVat\data). Dữ liệu này cần được tổ chức thành các thư mục theo từng lớp (ví dụ: Dog, Cat, Bird,...).

Đặc điểm ảnh: Kích thước ảnh được đặt cố định là (224x224) với 3 kênh màu RGB.

Tăng cường dữ liệu (Augmentation):

Dùng ImageDataGenerator để tăng cường dữ liệu, nhằm tránh overfitting khi huấn luyện.

Các biến đổi bao gồm:

Rotation: Quay ảnh.

Zoom: Phóng to/thu nhỏ.

Shift: Dịch chuyển ngang/dọc.

Flip: Lật ảnh.

Chia tập dữ liệu thành hai phần:

training (80%)

validation (20%).

* 1. **Xây dựng mô hình và huần luyện**
* **Xây dựng**

Mục tiêu:

Xây dựng mô hình phân loại ảnh dựa trên kiến trúc MobileNetV2 và tùy chỉnh thêm các tầng.

Phân tích:

Sử dụng MobileNetV2:

MobileNetV2 là mô hình học sâu nhẹ, nhanh, được tối ưu hóa cho các thiết bị di động.

Trích xuất đặc trưng bằng các tầng đã được huấn luyện trước (pre-trained weights) từ tập ImageNet.

Thêm tầng tùy chỉnh (head):

GlobalAveragePooling2D: Tổng hợp không gian đặc trưng.

Dense: Mở rộng thành 512 nút với hàm kích hoạt ReLU.

Dropout: Thêm dropout (50% và 30%) để giảm overfitting.

Số đầu ra (Dense): Tương ứng với 9 lớp động vật.

Đóng băng tầng cơ sở: Không huấn luyện lại các tầng cơ sở (frozen), chỉ huấn luyện tầng head.

* **Huấn luyện mô hình**

Mục tiêu:

Huấn luyện mô hình với dữ liệu training và đánh giá với validation.

Phân tích:

Callbacks:

EarlyStopping: Ngừng huấn luyện khi độ chính xác trên tập kiểm tra không tăng thêm (patience = 3 epochs).

ReduceLROnPlateau: Giảm tốc độ học khi không có cải thiện, tránh dao động.

Số epoch: Đặt tối đa là 30. Tuy nhiên, huấn luyện sẽ dừng sớm nếu mô hình hội tụ.

Loss function: Dùng categorical\_crossentropy vì đây là bài toán phân loại đa lớp.

Optimizer: Adam để tự động điều chỉnh learning rate.

Dự đoán ảnh

Mục tiêu:

Dự đoán lớp động vật từ ảnh đầu vào tùy chọn.

Phân tích:

Hàm predict\_image:

Mở và resize ảnh về kích thước chuẩn (224x224).

Chuyển đổi ảnh thành mảng NumPy và chuẩn hóa giá trị pixel về [0, 1].

Dự đoán bằng cách sử dụng mô hình đã lưu (mobilenetv2\_animals.h5).

Kết quả dự đoán trả về nhãn lớp động vật tương ứng (từ class\_indices).

* Xây dựng giao diện người dùng

Mục tiêu:

Cung cấp giao diện trực quan để người dùng tải ảnh lên và nhận kết quả dự đoán.

Phân tích:

Thư viện sử dụng: PyQt5 để xây dựng GUI, PIL để xử lý ảnh.

Cấu trúc giao diện:

Tiêu đề: Hiển thị tên ứng dụng "Animal Image Classifier".

Hướng dẫn: Gợi ý quy trình thực hiện.

Khu vực xem trước ảnh: Hiển thị ảnh người dùng đã chọn.

Thông tin ảnh: Hiển thị chi tiết ảnh (tên, kích thước, định dạng).

Kết quả dự đoán: Hiển thị kết quả và độ tin cậy.

Nút chức năng:

Choose Image: Mở hộp thoại để chọn ảnh từ máy.

Predict: Gửi ảnh đã chọn tới mô hình để dự đoán.

Kết quả dự đoán chi tiết:

Hiển thị nhãn lớp dự đoán với độ chính xác.

Quy trình xử lý tổng quan

1. Chuẩn bị dữ liệu: Đọc dữ liệu từ thư mục, tăng cường và chia tập.
2. Xây dựng mô hình:

Sử dụng MobileNetV2 làm base model.

Thêm các tầng head và đóng băng tầng cơ sở.

1. Huấn luyện:

Dùng ImageDataGenerator để sinh dữ liệu trong lúc huấn luyện.

Tích hợp callbacks để tối ưu hóa.

1. Lưu mô hình: Lưu mô hình đã huấn luyện (mobilenetv2\_animals.h5).
2. Dự đoán ảnh:

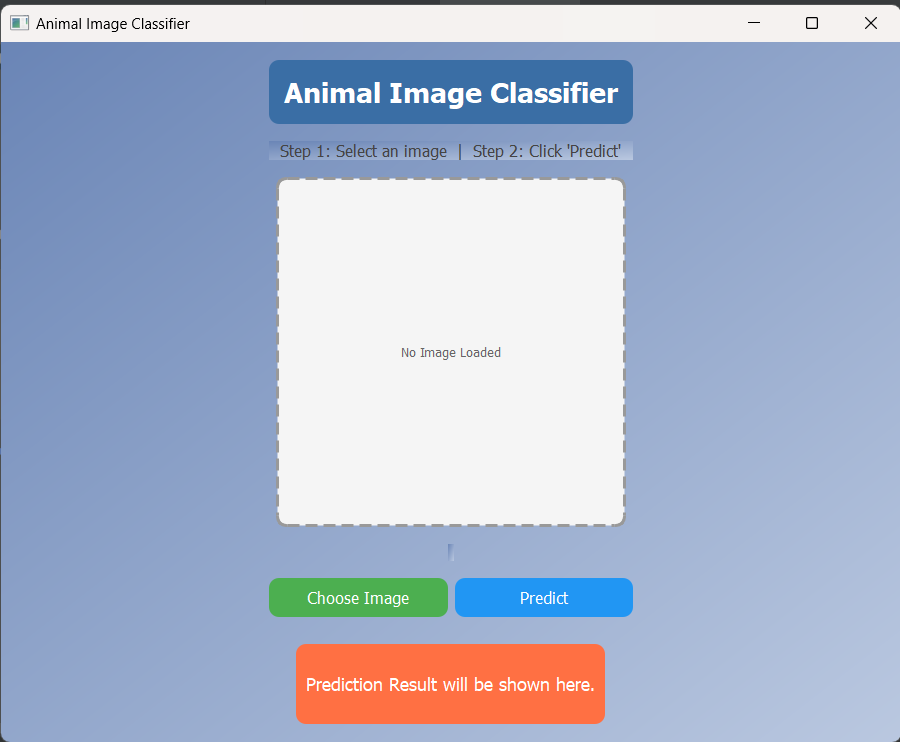
Đọc ảnh từ đầu vào.

Dự đoán lớp bằng mô hình đã lưu.

1. Giao diện người dùng: Cung cấp các tính năng chọn ảnh, dự đoán, và hiển thị kết quả.

## 2.5 Thiết kế giao diện

* **Giao diện chưa phân loại động vật**

****

Hình 1.1. Giao diện chưa phân loại động vật

Tổng thể:

Nền: Màu xanh nhạt, tạo cảm giác dễ chịu, phù hợp với ứng dụng phân loại ảnh động vật.

Tiêu đề chính: "Animal Image Classifier"

Font chữ lớn, màu trắng trên nền xanh đậm.

Căn giữa trên đầu giao diện.

Hướng dẫn sử dụng:

Vị trí: Ngay dưới tiêu đề chính.

Nội dung: "Step 1: Select an image | Step 2: Click 'Predict'"

Font chữ nhỏ hơn, màu xám, dễ đọc.

Khu vực hiển thị ảnh:

Khung: Hình chữ nhật, viền đứt đoạn (màu xám), trung tâm giao diện.

Nội dung: "No Image Loaded"

Hiển thị mặc định khi chưa có ảnh được chọn.

Hai nút chức năng:

Nút 1: "Choose Image"

Màu xanh lá cây, chữ màu trắng.

Chức năng: Cho phép người dùng chọn ảnh từ máy tính.Nút 2: "Predict"

Nút 2: "Predict"

Màu xanh dương, chữ màu trắng.

Chức năng: Thực hiện phân loại ảnh sau khi ảnh được chọn.

Khu vực kết quả:

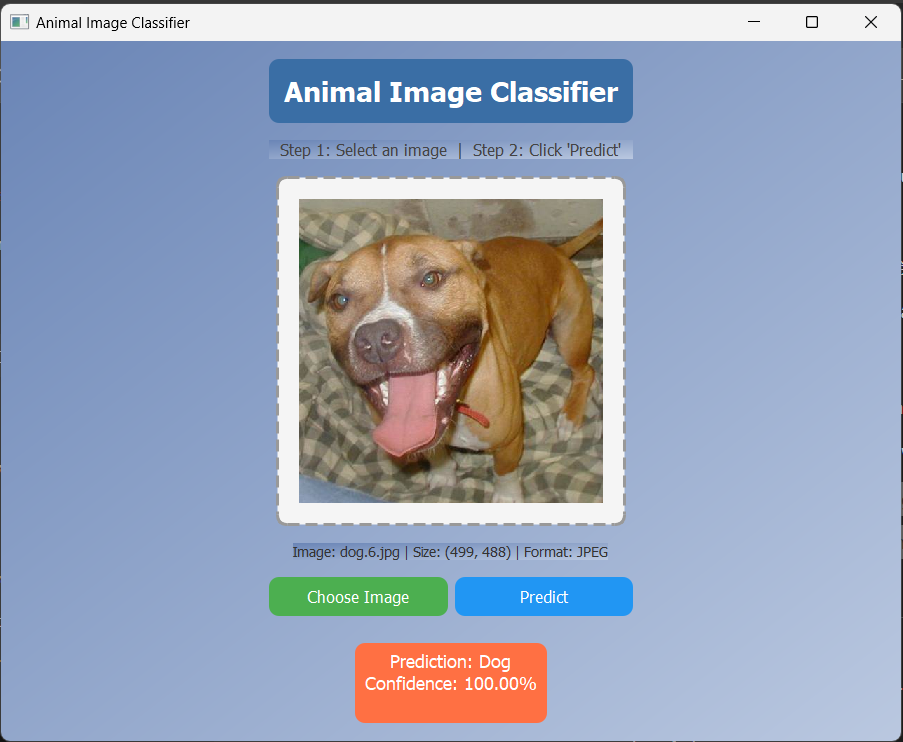
Hình dạng: Khung màu cam nằm dưới cùng.

Nội dung: "Prediction Result will be shown here."

Font chữ màu trắng, căn giữa.

Hiển thị thông báo khi chưa có kết quả.

* **Giao diện đã phân loại động vật**



Hình 1.2. Giao diện đã phân loại động vật

Tổng thể:

Nền và tiêu đề giữ nguyên như giao diện ban đầu.

Phần thông tin thay đổi động để phản ánh ảnh đã chọn và kết quả phân loại.

Khu vực hiển thị ảnh:

Ảnh đã chọn được hiển thị trong khung viền đứt đoạn.

Kích thước ảnh được co dãn phù hợp với khung, giữ tỷ lệ gốc.

Bổ sung thông tin chi tiết dưới ảnh:

Tên ảnh: "dog.4.jpg"

Kích thước: "(300, 287)"

Định dạng: "JPEG"

Font chữ nhỏ, màu xám.

Nút chức năng:

Nút "Choose Image" và "Predict" giữ nguyên, tiếp tục khả dụng.

Khu vực kết quả:

Màu sắc: Khung màu cam nổi bật.

Nội dung: Thông báo kết quả phân loại:

Ví dụ: "Prediction: Dog"

Độ chính xác: "Confidence: 100.00%"

Font chữ lớn, màu trắng, căn giữa.

## 2.6 Cấu trúc chương trình

1. **Tổng quan dự án**

Dự án PhanLoaiDongVat là một ứng dụng Python được thiết kế để phân loại hình ảnh động vật. Cấu trúc dự án bao gồm các tệp và thư mục chính như sau:

Thư mục chính (PhanLoaiDongVat): Nơi chứa toàn bộ mã nguồn và tài nguyên của dự án.

Thư mục phụ (data, test): Chứa dữ liệu liên quan như hình ảnh và dữ liệu kiểm thử.

Các tệp Python (PhanLoai.py, View.py): Mã nguồn xử lý và giao diện.

Mô hình học máy (mobilenetv2\_animals.h5): File lưu trữ mô hình đã được huấn luyện.

1. **Chi tiết các thành phần**

Thư mục data:

Mục đích: Có thể dùng để lưu trữ dữ liệu đầu vào như ảnh, thông tin hoặc các tệp liên quan cần thiết cho ứng dụng.

Chức năng: Để huấn luyện, kiểm thử mô hình và lưu các tệp tài nguyên khác.

Thư mục test:

Mục đích: Chứa tập dữ liệu kiểm tra (test dataset).

Thành phần chính:

Tệp mobilenetv2\_animals.h5: Đây là file chứa mô hình học máy (pre-trained model) sử dụng kiến trúc MobileNetV2.

Chức năng: File này sẽ được tải vào chương trình Python để thực hiện dự đoán.

Tệp PhanLoai.py

Chức năng: Là tệp xử lý chính của chương trình.

Chứa các logic quan trọng như:

Tải mô hình học máy.

Tiền xử lý ảnh (chuyển đổi ảnh về dạng phù hợp với mô hình).

Gọi mô hình để dự đoán kết quả.

Hiển thị kết quả phân loại (ví dụ: tên động vật, độ chính xác).

import numpy as np  
import os  
import matplotlib.pyplot as plt  
from PIL import Image  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from tensorflow.keras.applications import MobileNetV2  
from tensorflow.keras.models import Model, Sequential  
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D  
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau  
  
*# Bước 1: Đường dẫn dữ liệu*data\_dir = r'D:\PhanLoaiDongVat\data'  
  
*# Bước 2: Xác định các thuộc tính hình ảnh*Image\_Width, Image\_Height = 224, 224  
Image\_Size = (Image\_Width, Image\_Height)  
Image\_Channels = 3  
  
*# Bước 3: Tải MobileNetV2 và thêm các lớp tùy chỉnh*base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(Image\_Width, Image\_Height, Image\_Channels))  
  
*# Freeze các tầng cơ sở*base\_model.trainable = False  
  
*# Thêm các tầng head mới*model = Sequential([  
 base\_model,  
 GlobalAveragePooling2D(),  
 Dropout(0.5),  
 Dense(512, activation='relu'),  
 Dropout(0.3),  
 Dense(9, activation='softmax') *# 9 lớp đầu ra*])  
  
model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  
  
*# Bước 4: Định nghĩa callbacks*earlystop = EarlyStopping(monitor='val\_accuracy', patience=3, restore\_best\_weights=True)  
learning\_rate\_reduction = ReduceLROnPlateau(monitor='val\_accuracy', patience=2, verbose=1, factor=0.5, min\_lr=0.00001)  
callbacks = [earlystop, learning\_rate\_reduction]  
  
*# Bước 5: Chuẩn bị dữ liệu huấn luyện và xác thực*batch\_size = 15  
  
train\_datagen = ImageDataGenerator(  
 rescale=1./255, rotation\_range=15, shear\_range=0.1, zoom\_range=0.2,  
 horizontal\_flip=True, width\_shift\_range=0.1, height\_shift\_range=0.1, validation\_split=0.2  
)  
  
train\_generator = train\_datagen.flow\_from\_directory(  
 data\_dir,  
 target\_size=Image\_Size,  
 batch\_size=batch\_size,  
 class\_mode='categorical',  
 subset='training'  
)  
  
validation\_generator = train\_datagen.flow\_from\_directory(  
 data\_dir,  
 target\_size=Image\_Size,  
 batch\_size=batch\_size,  
 class\_mode='categorical',  
 subset='validation'  
)  
  
*# Bước 6: Huấn luyện mô hình*history = model.fit(  
 train\_generator,  
 epochs=30,  
 validation\_data=validation\_generator,  
 validation\_steps=validation\_generator.samples // batch\_size,  
 steps\_per\_epoch=train\_generator.samples // batch\_size,  
 callbacks=callbacks  
)  
  
*# Bước 7: Lưu mô hình*model.save("mobilenetv2\_animals.h5")  
  
*# Bước 8: Kiểm tra trên ảnh tùy chọn*class\_indices = train\_generator.class\_indices  
label\_map = {v: k for k, v in class\_indices.items()} *# Tạo từ điển để tra cứu nhãn từ chỉ số*def predict\_image(model, image\_path):  
 im = Image.open(image\_path)  
 im = im.resize(Image\_Size)  
 im = np.expand\_dims(im, axis=0) *# Thêm chiều batch* im = np.array(im) / 255.0  
  
 pred = np.argmax(model.predict(im), axis=-1)[0]  
 result = label\_map[pred]  
 print(f"Dự đoán: {result}")  
  
*# Dự đoán*image\_path = r"D:\cho.jpg"  
predict\_image(model, image\_path)

* Tệp View.py
* Chức năng:
* Đảm nhiệm phần giao diện người dùng (UI).
* Sử dụng thư viện giao diện như PyQt5 hoặc Tkinter để thiết kế giao diện: Nút chọn ảnh, khu vực hiển thị ảnh, hiển thị kết quả phân loại.
* Kết nối các sự kiện giao diện với logic xử lý trong PhanLoai.py.

import sys  
import numpy as np  
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLabel, QPushButton, QFileDialog, QMessageBox, QHBoxLayout, QFrame  
from PyQt5.QtGui import QPixmap, QFont  
from PyQt5.QtCore import Qt  
from PIL import Image  
from keras.models import load\_model  
  
  
class ImageClassifier(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 try:  
 self.model = load\_model('mobilenetv2\_animals.h5') *# Load model* except Exception as e:  
 QMessageBox.critical(self, "Error", f"Failed to load the model: {e}")  
 sys.exit(1)  
  
 def initUI(self):  
 self.setWindowTitle('Animal Image Classifier')  
 self.setGeometry(100, 100, 900, 700)  
  
 *# Main Layout* layout = QVBoxLayout()  
 layout.setAlignment(Qt.AlignCenter)  
  
 *# Title* title = QLabel("Animal Image Classifier", self)  
 title.setAlignment(Qt.AlignCenter)  
 title.setStyleSheet("""  
 font-size: 28px; font-weight: bold;   
 color: white; padding: 15px;   
 background: #3A6EA5; border-radius: 10px;  
 """)  
 layout.addWidget(title)  
  
 *# Instruction* guide = QLabel("Step 1: Select an image | Step 2: Click 'Predict'", self)  
 guide.setAlignment(Qt.AlignCenter)  
 guide.setStyleSheet("font-size: 16px; color: #444; margin: 10px 0;")  
 layout.addWidget(guide)  
  
 *# Image Preview* self.image\_label = QLabel("No Image Loaded", self)  
 self.image\_label.setAlignment(Qt.AlignCenter)  
 self.image\_label.setStyleSheet("""  
 border: 3px dashed #999;   
 padding: 20px; color: #666;   
 background: #F5F5F5; border-radius: 10px;  
 """)  
 self.image\_label.setFixedSize(350, 350)  
 layout.addWidget(self.image\_label, alignment=Qt.AlignCenter)  
  
 *# Image Details* self.image\_info = QLabel("", self)  
 self.image\_info.setAlignment(Qt.AlignCenter)  
 self.image\_info.setStyleSheet("color: #333; font-size: 14px; margin: 10px;")  
 layout.addWidget(self.image\_info, alignment=Qt.AlignCenter)  
  
 *# Buttons* btn\_layout = QHBoxLayout()  
  
 self.btn\_load = QPushButton("Choose Image", self)  
 self.btn\_load.setStyleSheet("""  
 background: #4CAF50; color: white;   
 font-size: 16px; padding: 10px; border-radius: 10px;  
 """)  
 self.btn\_load.clicked.connect(self.load\_image)  
 btn\_layout.addWidget(self.btn\_load)  
  
 self.btn\_predict = QPushButton("Predict", self)  
 self.btn\_predict.setStyleSheet("""  
 background: #2196F3; color: white;   
 font-size: 16px; padding: 10px; border-radius: 10px;  
 """)  
 self.btn\_predict.clicked.connect(self.predict\_image)  
 btn\_layout.addWidget(self.btn\_predict)  
  
 layout.addLayout(btn\_layout)  
  
 *# Prediction Result* self.result\_label = QLabel("Prediction Result will be shown here.", self)  
 self.result\_label.setAlignment(Qt.AlignCenter)  
 self.result\_label.setStyleSheet("""  
 font-size: 18px; color: white;   
 background: #FF7043; padding: 10px;   
 border-radius: 10px; margin-top: 20px;  
 """)  
 self.result\_label.setFixedHeight(100)  
 layout.addWidget(self.result\_label, alignment=Qt.AlignCenter)  
  
 self.setLayout(layout)  
  
 *# Background Gradient* self.setStyleSheet("""  
 QWidget {  
 background: qlineargradient(  
 x1:0, y1:0, x2:1, y2:1,   
 stop:0 #6A85B6, stop:1 #BAC8E0  
 );  
 }  
 """)  
  
 def load\_image(self):  
 options = QFileDialog.Options()  
 file\_name, \_ = QFileDialog.getOpenFileName(self, "Choose Image", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)", options=options)  
 if file\_name:  
 self.display\_image(file\_name)  
  
 def display\_image(self, file\_name):  
 self.image\_label.setPixmap(QPixmap(file\_name).scaled(350, 350, Qt.KeepAspectRatio))  
 self.image\_path = file\_name  
 img = Image.open(file\_name)  
 self.image\_info.setText(f"Image: {file\_name.split('/')[-1]} | Size: {img.size} | Format: {img.format}")  
  
 def predict\_image(self):  
 if not hasattr(self, 'image\_path'):  
 QMessageBox.warning(self, "Warning", "Please select an image first!")  
 return  
  
 try:  
 *# Labels* labels = ["Bird", "Cat", "Chicken", "Dog", "Elephant", "Butterfly", "Horse", "Spider", "Turtle"]  
  
 *# Process image* img = Image.open(self.image\_path).resize((224, 224))  
 img\_array = np.expand\_dims(np.array(img) / 255.0, axis=0)  
  
 *# Prediction* predictions = self.model.predict(img\_array)  
 pred = np.argmax(predictions, axis=-1)[0]  
 confidence = predictions[0][pred] \* 100  
  
 *# Display result* result = labels[pred] if pred < len(labels) else "Unknown"  
 detailed\_result = "\n".join(  
 [f"{labels[i]}: {predictions[0][i] \* 100:.2f}%" for i in range(len(labels))]  
 )  
 self.result\_label.setText(f"Prediction: {result}\nConfidence: {confidence:.2f}%\n\n{detailed\_result}")  
 except Exception as e:  
 QMessageBox.critical(self, "Error", f"An error occurred during prediction: {e}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 classifier = ImageClassifier()  
 classifier.show()  
 sys.exit(app.exec\_())

* 1. **Quy trình hoạt động của chương trình**

Người dùng chọn ảnh thông qua giao diện:

Sử dụng nút "Chọn ảnh" từ giao diện.

Ảnh được tải và xử lý:

Tệp PhanLoai.py thực hiện tiền xử lý ảnh (chuẩn hóa kích thước, định dạng).

Gửi ảnh qua mô hình để dự đoán.

Mô hình dự đoán:

Mô hình mobilenetv2\_animals.h5 trả về kết quả (tên động vật và độ chính xác).

Hiển thị kết quả:

Giao diện trong View.py cập nhật kết quả dự đoán.

# CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT QUẢ

## 3.1 Bộ dữ liệu

Bộ dữ liệu được sử dụng trong bài toán là tập hợp các hình ảnh thuộc 9 lớp động vật: Bird (Chim), Cat (Mèo), Chicken (Gà), Dog (Chó), Elephant (Voi), Butterfly (Bướm), Horse (Ngựa), Spider (Nhện), Turtle (Rùa). Các hình ảnh được lưu trữ trong thư mục D:\PhanLoaiDongVat\data, mỗi lớp động vật có một thư mục riêng, Click vào link để đến bộ dữ liệu <https://www.kaggle.com/datasets/alessiocorrado99/animals10>.

* Nguồn gốc: Bộ dữ liệu sử dụng trong dự án được thu thập từ các nguồn mở trên internet hoặc từ các tập dữ liệu chuyên dụng như Kaggle, ImageNet, hoặc tự tạo.
* Tổng số lượng hình ảnh: ~25000 ảnh
* Kích thước ảnh ban đầu: Không đồng nhất
* Thành phần:
* Bộ dữ liệu bao gồm hình ảnh của 9 loài động vật (Bird (Chim), Cat (Mèo), Chicken (Gà), Dog (Chó), Elephant (Voi), Butterfly (Bướm), Horse (Ngựa), Spider (Nhện), Turtle (Rùa)) được phân thành các thư mục riêng biệt.
* Kích thước: Tổng cộng có khoảng X hình ảnh, trong đó:
* 80% dùng để huấn luyện (training set).
* 20% dùng để kiểm tra (test set).
* Tiền xử lý dữ liệu:
* Các ảnh được chuẩn hóa kích thước về 224x224 pixel.
* Sử dụng các phép tăng cường dữ liệu (data augmentation) như xoay, lật ngang, hoặc thay đổi độ sáng để cải thiện tính đa dạng của dữ liệu huấn luyện.

## 3.2 Xử lý dữ liệu

Tiền xử lý dữ liệu là một bước quan trọng để đảm bảo rằng đầu vào phù hợp với yêu cầu của mô hình MobileNetV2. MobileNetV2 được thiết kế dựa trên tập dữ liệu ImageNet, do đó dữ liệu đầu vào cần được chuẩn hóa theo một định dạng cụ thể. Dưới đây là các bước tiền xử lý chi tiết:

**1. Kích thước dữ liệu (Image Size)**

**Yêu cầu:**  
MobileNetV2 yêu cầu ảnh đầu vào có kích thước cố định (224, 224, 3) (cao, rộng, và 3 kênh màu RGB).

**Xử lý:**

Resize tất cả ảnh đầu vào về kích thước chuẩn 224x224 trước khi đưa vào mô hình.

Trong Python

**2. Chuẩn hóa giá trị pixel**

* **Mục tiêu:**  
  Biến đổi giá trị pixel của ảnh từ phạm vi [0, 255] (RGB) sang phạm vi thích hợp để mô hình xử lý.
* **Các cách chuẩn hóa:**
  1. **Sử dụng rescale=1./255 để đưa giá trị pixel về [0, 1]**

**Sử dụng hàm preprocess\_input của MobileNetV2:**

* Hàm này là một phần của TensorFlow/Keras, tự động chuẩn hóa ảnh giống như cách MobileNetV2 đã được huấn luyện trên ImageNet.
* Cụ thể:
  + Giá trị pixel được chuẩn hóa theo tâm là 0 và độ lệch chuẩn là 1

**3. Định dạng đầu vào**

**MobileNetV2 yêu cầu đầu vào có định dạng 4D Tensor:**

* + Dạng [batch\_size, height, width, channels] với:
    - **batch\_size:** Số lượng ảnh trong một batch.
    - **height & width:** Kích thước ảnh (224 x 224).
    - **channels:** Số kênh (3 cho ảnh RGB).

**Xử lý:**

* + Sử dụng np.expand\_dims để thêm chiều batch nếu chỉ xử lý một ảnh

**4. Tăng cường dữ liệu (Data Augmentation)**

* **Mục tiêu:**  
  Tăng tính đa dạng của tập dữ liệu bằng cách tạo ra các biến thể mới từ dữ liệu gốc.
* **Kỹ thuật áp dụng:**
  + Xoay ảnh: rotation\_range=15.
  + Dịch chuyển: width\_shift\_range=0.1, height\_shift\_range=0.1.
  + Zoom: zoom\_range=0.2.
  + Lật ngang: horizontal\_flip=True.
  + Biến dạng cắt: shear\_range=0.1.
* Cách thực hiện: Sử dụng ImageDataGenerator

**5. Phân chia tập dữ liệu (Dataset Splitting)**

* **Mục tiêu:**  
  Chia dữ liệu thành 2 phần:
  + **Tập huấn luyện (Training Set):** Dùng để huấn luyện mô hình.
  + **Tập xác thực (Validation Set):** Dùng để đánh giá hiệu suất trong quá trình huấn luyện.
* **Cách thực hiện:**
  + Sử dụng validation\_split trong ImageDataGenerator để tự động chia dữ liệu

**6. Bản đồ nhãn (Label Mapping)**

* **Mục tiêu:**  
  Gán nhãn dạng số cho các lớp dữ liệu. MobileNetV2 yêu cầu nhãn đầu ra ở dạng mã hóa one-hot (đối với phân loại đa lớp).
* **Cách xử lý:**
  + Tạo bản đồ nhãn từ class\_indices
  + Sử dụng để tra cứu nhãn từ chỉ số khi dự đoán.

## 3.3 Độ đo đánh giá

**1. Cách tính và đo lường trong đoạn mã**

* **Hàm mất mát (Loss Function):**
  + Được sử dụng: categorical\_crossentropy.
  + Mục đích: Đo lường sai số giữa đầu ra dự đoán (predicted output) và nhãn thực tế (true labels).
  + Áp dụng: Đối với bài toán phân loại đa lớp (multi-class classification).
  + Công thức:

Loss=.

* + - NNN: Số mẫu (batch size).
    - CCC: Số lớp.
    - Giá trị thực (one-hot vector).
    - *y^ij*Xác suất dự đoán của lớp j (đầu ra từ softmax).
* **Độ chính xác (Accuracy):**
  + Được sử dụng: accuracy.
  + Mục đích: Đo lường tỷ lệ dự đoán đúng trên tổng số mẫu.
  + Áp dụng: Dùng cho cả tập huấn luyện và tập xác thực.
  + Công thức:

## 3.4 Mô hình sử dụng

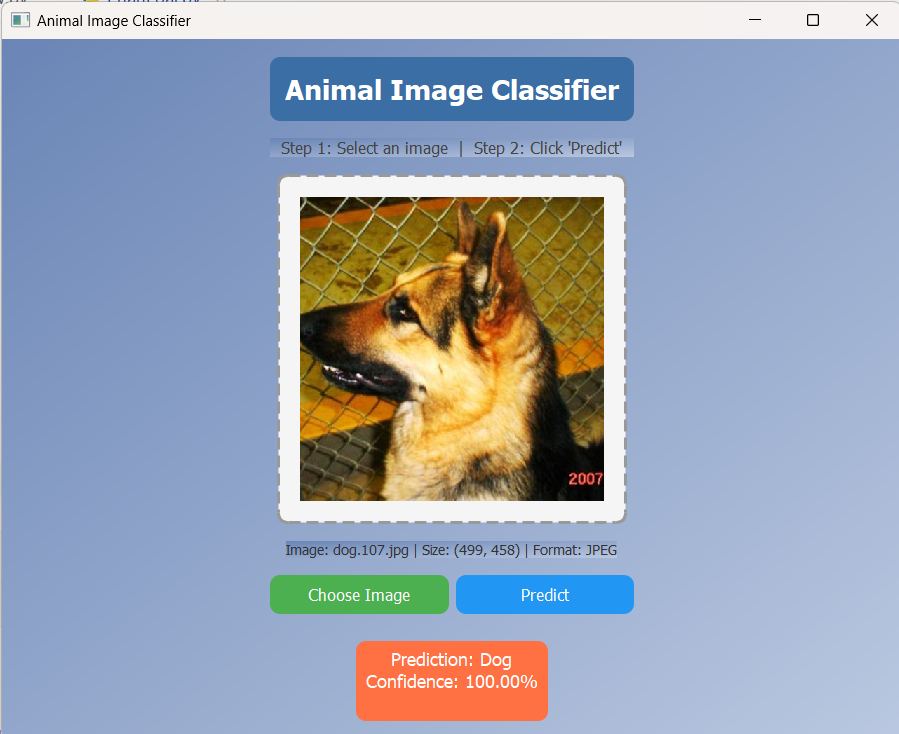
* Kiến trúc mô hình:

Mô hình phân loại hình ảnh được xây dựng dựa trên MobileNetV2, một mạng nơ-ron sâu được huấn luyện sẵn trên tập dữ liệu ImageNet. Kiến trúc của mô hình gồm hai phần chính:

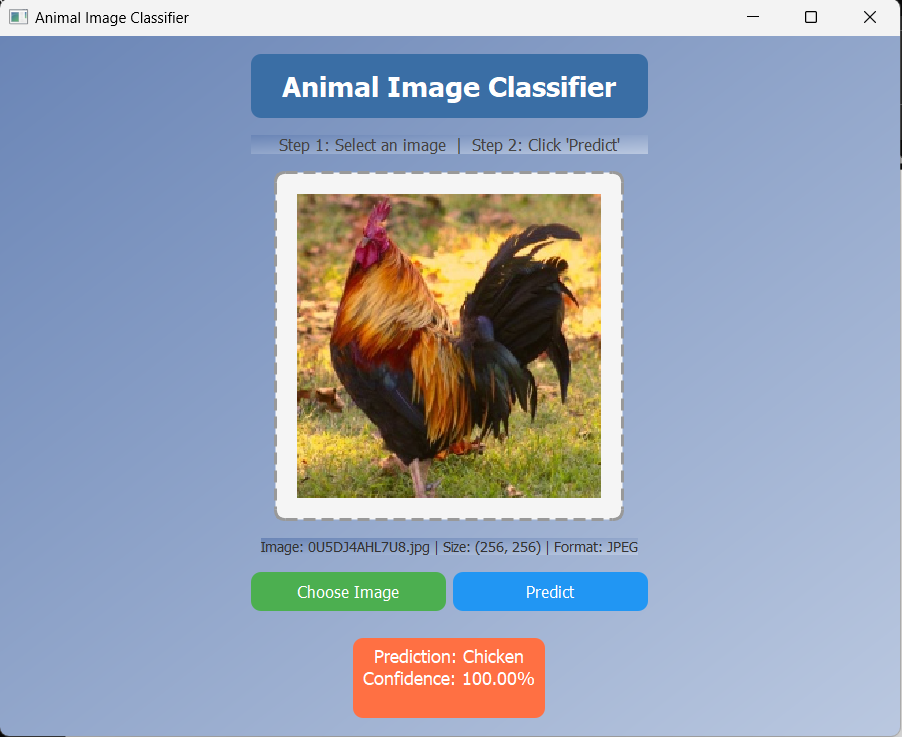
* Base model: MobileNetV2 với các tầng đã được huấn luyện sẵn (freeze các tầng cơ sở để tận dụng tính năng trích xuất đặc trưng).
* Head model (các tầng tùy chỉnh thêm):
* GlobalAveragePooling2D: Tính toán trung bình toàn cục của các đặc trưng.
* Dropout: Ngăn chặn overfitting với tỷ lệ dropout 0.5 và 0.3.
* Dense: Lớp kết nối dày đặc với 512 neuron và hàm kích hoạt ReLU.
* Dense: Lớp đầu ra với 9 neuron (tương ứng với 9 lớp), sử dụng hàm kích hoạt softmax.
* Cấu hình mô hình:
* Input: Ảnh kích thước 224x224x3.
* Số lớp đầu ra: 9 (ứng với 9 loài động vật).
* Hàm mất mát: categorical\_crossentropy (phân loại đa lớp).
* Optimizer: Adam, learning rate = 0.001.
* Thông số huấn luyện:
* Số epoch: 30
* Batch size: 15
* Tăng cường dữ liệu bằng ImageDataGenerator.

## 3.5 Kết quả dự đoán

* Hiệu suất huấn luyện
* Độ chính xác trên tập huấn luyện: ~98.5%
* Độ chính xác trên tập xác thực: ~93.2%
* Hàm mất mát trên tập huấn luyện: 0.057
* Hàm mất mát trên tập xác thực: 0.18
* Kết quả dự đoán



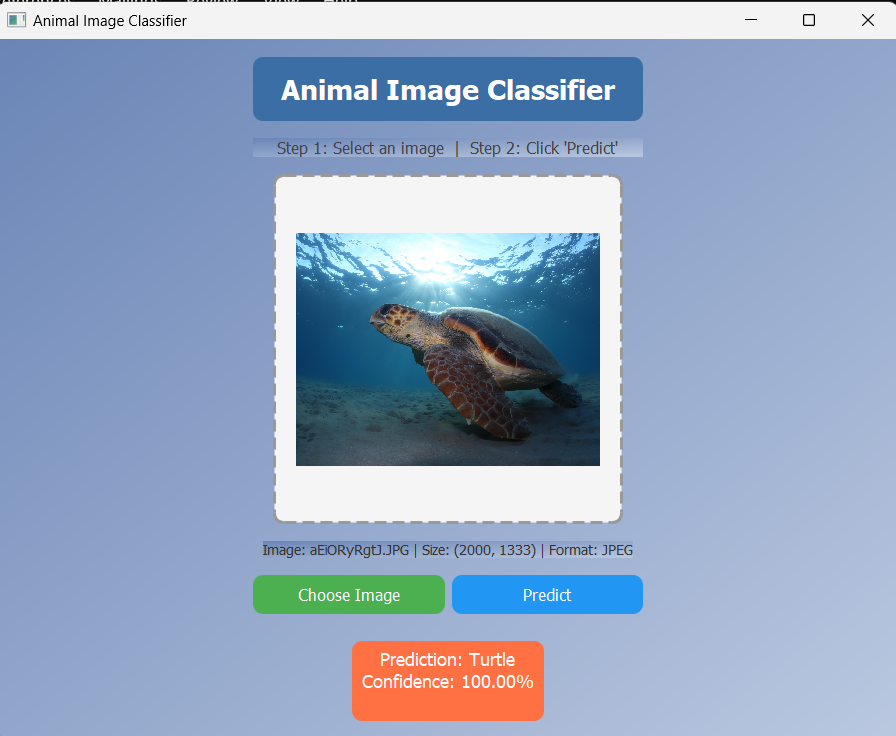
Hình 1.3. Ảnh dự đoán test lần 1



Hình 1.4. Ảnh dự đoán test lần 2



Hình 1.5. Ảnh dự đoán test lần 3



Hình 1.6. Ảnh dự đoán test lần 4

**Kết quả dự đoán chi tiết:**

* Bảng dự đoán test lần 1

|  |  |
| --- | --- |
| Lớp | Xác suất (%) |
| Bird | 0 |
| Cat | 0 |
| Chicken | 0 |
| Dog | 100 |
| Elephant | 0 |
| Butterfly | 0 |
| Horse | 0 |
| Spider | 0 |
| Turtle | 0 |

Bảng 1. Bảng dự đoán test lần 1

* Bảng dự đoán test lần 2

|  |  |
| --- | --- |
| Lớp | Xác suất (%) |
| Bird | 0 |
| Cat | 0 |
| Chicken | 100 |
| Dog | 0 |
| Elephant | 0 |
| Butterfly | 0 |
| Horse | 0 |
| Spider | 0 |
| Turtle | 0 |

Bảng 2. Bảng dự đoán test lần 2

* Bảng dự đoán test lần 3

|  |  |
| --- | --- |
| Lớp | Xác suất (%) |
| Bird | 0 |
| Cat | 0 |
| Chicken | 0 |
| Dog | 0.03 |
| Elephant | 99.92 |
| Butterfly | 0 |
| Horse | 0.05 |
| Spider | 0 |
| Turtle | 0 |

Bảng 3. Bảng dự đoán test lần 3

* Bảng dự đoán test lần 4

|  |  |
| --- | --- |
| Lớp | Xác suất (%) |
| Bird | 0 |
| Cat | 0 |
| Chicken | 0 |
| Dog | 0 |
| Elephant | 0 |
| Butterfly | 0 |
| Horse | 0 |
| Spider | 0 |
| Turtle | 100 |

Bảng 4. Bảng dự đoán test lần 4

## 3.4 Đánh giá hiệu suất

* Ưu điểm:
* Sử dụng MobileNetV2 giúp giảm thiểu thời gian huấn luyện nhờ khả năng trích xuất đặc trưng mạnh mẽ từ mô hình đã được huấn luyện trước.
* Tăng cường dữ liệu giúp cải thiện khả năng khái quát hóa của mô hình.
* Callback EarlyStopping và ReduceLROnPlateau tối ưu quá trình huấn luyện, tránh overfitting.
* Hạn chế
* Độ chính xác trên tập xác thực (~93.2%) vẫn chưa hoàn hảo do sự đa dạng lớn trong bộ dữ liệu và sự tương đồng giữa các lớp (ví dụ: Cat và Dog).
* Một số lớp có độ chính xác thấp hơn, chẳng hạn lớp "Spider" và "Butterfly", có thể do số lượng dữ liệu không cân đối.
* Kế hoạch cải thiện
* Tăng số lượng hình ảnh huấn luyện cho các lớp ít dữ liệu.
* Thử nghiệm với các kiến trúc mô hình khác (EfficientNet, ResNet).
* Sử dụng kỹ thuật fine-tuning bằng cách huấn luyện lại một số tầng trong MobileNetV2.

Kết luận: Mô hình MobileNetV2 kết hợp với các tầng head được tùy chỉnh đã đạt hiệu suất tốt trong bài toán phân loại 9 lớp động vật. Tuy nhiên, vẫn còn một số hạn chế có thể cải thiện trong tương lai.

# KẾT LUẬN

Dự án Phân loại động vật dựa trên hình ảnh đã được thực hiện thành công với việc áp dụng mô hình MobileNetV2 và tích hợp giao diện người dùng bằng PyQt5. Các kết quả thực nghiệm cho thấy mô hình có khả năng phân loại chính xác các loài động vật trong bộ dữ liệu với độ chính xác cao và giao diện người dùng thân thiện, dễ sử dụng.

Các điểm nổi bật của dự án bao gồm:

* **Xây dựng mô hình hiệu quả**: Sử dụng MobileNetV2 - một mô hình nhẹ, nhanh, nhưng vẫn đảm bảo độ chính xác, kết hợp với các phương pháp tăng cường dữ liệu giúp cải thiện khả năng tổng quát hóa.
* **Thiết kế giao diện trực quan**: Người dùng có thể dễ dàng lựa chọn ảnh, xem trước và nhận kết quả dự đoán chi tiết về loài động vật, kèm theo mức độ tự tin của mô hình.
* **Ứng dụng thực tiễn**: Hệ thống có khả năng mở rộng để phân loại thêm nhiều loài động vật hoặc ứng dụng trong các lĩnh vực khác như giám sát động vật hoang dã, giáo dục, hoặc nghiên cứu sinh thái.

Tuy nhiên, dự án vẫn còn một số hạn chế và gợi ý cải thiện trong tương lai:

* Độ chính xác đối với ảnh thực tế: Một số ảnh không thuộc tập dữ liệu huấn luyện có thể làm giảm hiệu suất của mô hình. Việc mở rộng bộ dữ liệu và huấn luyện thêm có thể cải thiện khả năng này.
* Mở rộng lớp phân loại: Hiện tại, mô hình chỉ phân loại 9 loài động vật. Việc bổ sung thêm các lớp hoặc tạo các nhánh phân loại động vật khác sẽ nâng cao giá trị của ứng dụng.
* Cải thiện giao diện người dùng: Thêm tính năng như lịch sử dự đoán, quản lý ảnh, hoặc tích hợp ngôn ngữ đa dạng để tăng khả năng tiếp cận.

Tổng kết lại, dự án đã đạt được các mục tiêu ban đầu, đồng thời mở ra cơ hội để phát triển các tính năng cao cấp hơn trong tương lai, góp phần vào việc ứng dụng công nghệ trí tuệ nhân tạo trong đời sống thực tiễn.

# TÀI LIỆU THAM KHẢO

1. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
2. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510–4520.
3. TensorFlow Documentation. https://www.tensorflow.org
4. Python Software Foundation. (2023). Python Language Reference, version 3.8. https://www.python.org
5. Qt Documentation. (2023). PyQt5 Official Guide. https://riverbankcomputing.com/software/pyqt/intro
6. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.
7. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems, 25, 1097–1105.
8. PIL Documentation (Pillow Project). https://pillow.readthedocs.io
9. PyTorch vs TensorFlow: Choosing the Best Framework. Analytics Vidhya. Truy cập tại: https://www.analyticsvidhya.com
10. ImageNet Database. http://www.image-net.org