**HỌC VIỆN NGÂN HÀNG**



**KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: ỨNG DỤNG COMPUTER VISION TỰ ĐỘNG NHẬN DIỆN ĐỘNG VẬT**

**Giảng viên hướng dẫn: TS Vũ Trọng Sinh**

**Nhóm sinh viên thực hiện: Nhóm 7**

***Hà Nội – 6/2025***

**HỌC VIỆN NGÂN HÀNG**

**KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: ỨNG DỤNG COMPUTER VISION TỰ ĐỘNG NHẬN DIỆN ĐỘNG VẬT**

**Giảng viên hướng dẫn: TS Vũ Trọng Sinh**

**Sinh viên thực hiện:       Vũ Thị Minh Phương**

**Nguyễn Chí Bách**

**Đặng Ngọc Ánh**

**Nguyễn Đức Tâm**

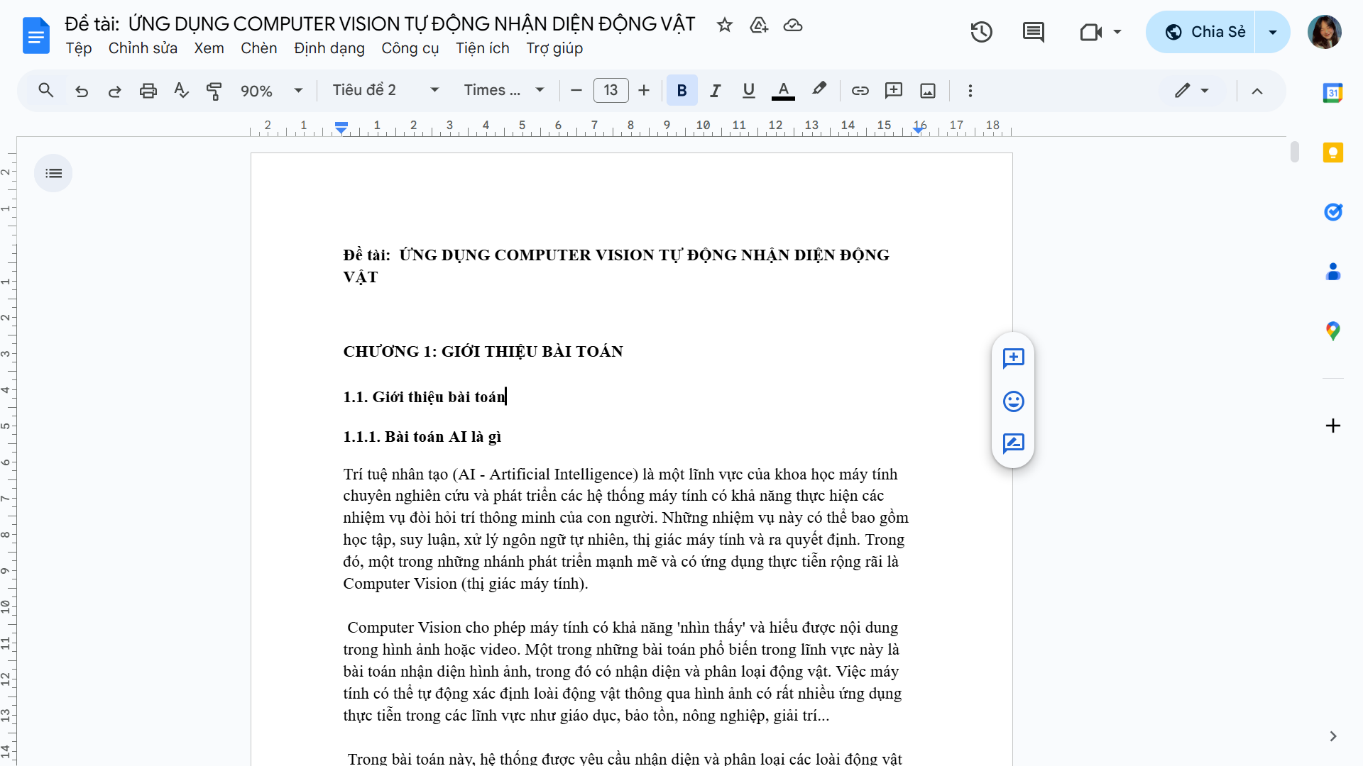
***Hà Nội - 6/2025***

**BẢNG PHÂN CÔNG CÔNG VIỆC**

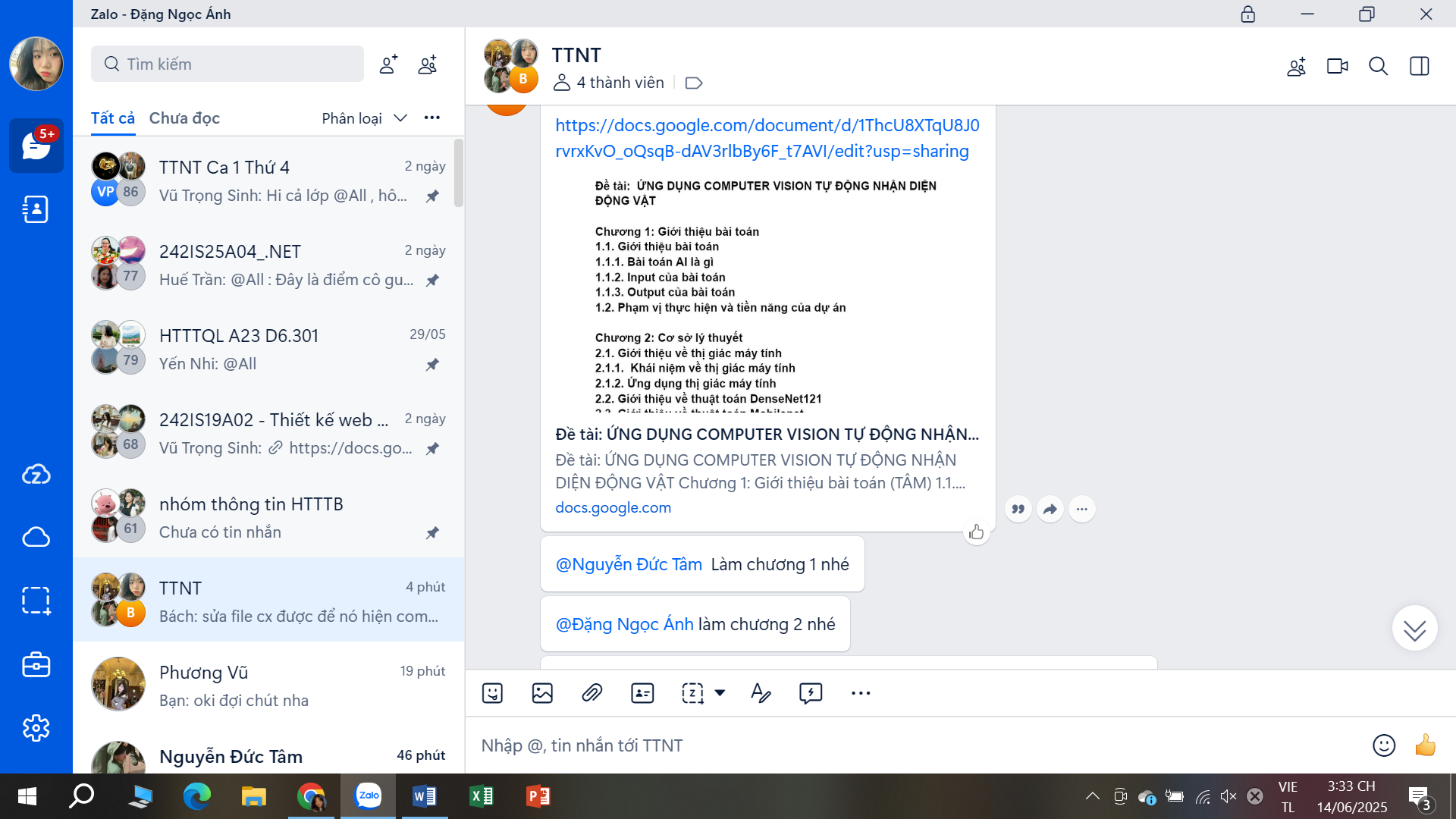
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Họ Và Tên** | **MSV** | **Phân Công** | **% Đóng Góp** | **Chữ Ký** |
| **1** | Vũ Thị Minh Phương | 25A4041891 |  |  |  |
| **2** | Nguyễn Chí Bách | 25A4041489 |  |  |  |
| **3** | Đặng Ngọc Ánh | 25A4041479 |  |  |  |
| **4** | Nguyễn Đức Tâm | 25A4041915 |  |  |  |

**MINH CHỨNG CÔNG VIỆC**

**Google docs**



**Zalo**



**LỜI CẢM ƠN**

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến Khoa Công nghệ thông tin & Kinh tế số - Học viện Ngân hàng đã tạo điều kiện thuận lợi để chúng em được học tập và thực hiện đề tài **“Ứng dụng Computer Vision tự động nhận diện động vật”** trong khuôn khổ học phần **Trí tuệ nhân tạo**.

Đặc biệt, nhóm xin bày tỏ lòng biết ơn sâu sắc đến **Thầy Vũ Trọng Sinh** – giảng viên phụ trách môn học, người đã tận tình giảng dạy và truyền đạt cho chúng em những kiến thức quý báu, hiện đại và có tính ứng dụng cao trong lĩnh vực trí tuệ nhân tạo nói chung và thị giác máy tính nói riêng. Sự hướng dẫn tận tâm và những chia sẻ thực tiễn từ Thầy đã giúp chúng em định hướng rõ ràng trong suốt quá trình triển khai đề tài.

Môn Trí tuệ nhân tạo là một học phần thiết thực và bổ ích, không chỉ giúp sinh viên tiếp cận gần hơn với các công nghệ tiên tiến mà còn rèn luyện kỹ năng phân tích, tư duy logic và khả năng vận dụng kiến thức vào thực tiễn. Trong quá trình thực hiện đề tài, tuy đã cố gắng nỗ lực hết mình, nhưng nhóm vẫn khó tránh khỏi những thiếu sót. Kính mong Thầy xem xét và góp ý để nhóm có thể hoàn thiện đề tài một cách tốt hơn.

Nhóm xin chân thành cảm ơn!

**Nhóm sinh viên thực hiện**

**LỜI CAM ĐOAN**

Đề tài **“Ứng dụng Computer Vision tự động nhận diện động vật”** là kết quả của quá trình nghiên cứu, tìm hiểu và thực hiện nghiêm túc của toàn bộ thành viên trong nhóm, dưới sự hướng dẫn tận tình của **Thầy Vũ Trọng Sinh** - giảng viên phụ trách môn **Trí tuệ nhân tạo**.

Trong suốt quá trình thực hiện, nhóm đã chủ động tìm hiểu lý thuyết, áp dụng kiến thức được học vào thực tiễn, đồng thời tham khảo có chọn lọc các tài liệu liên quan để hoàn thiện đề tài. Các tài liệu tham khảo, hình ảnh sử dụng trong đề tài đều được trích dẫn đầy đủ và rõ ràng theo đúng quy định.

Nhóm hoàn toàn chịu trách nhiệm trước Nhà trường và Giảng viên hướng dẫn về tính trung thực, khách quan và bản quyền của toàn bộ nội dung được trình bày trong báo cáo.

Nhóm xin chân thành cảm ơn sự hỗ trợ và hướng dẫn quý báu của Thầy trong suốt quá trình thực hiện đề tài.

**Nhóm sinh viên thực hiện**

**MỤC LỤC**

[**LỜI CẢM ƠN** 1](#_Toc200808795)

[**LỜI CAM ĐOAN** 2](#_Toc200808796)

[**DANH MỤC HÌNH ẢNH** 5](#_Toc200808797)

[**DANH MỤC BẢNG BIỂU** 5](#_Toc200808798)

[**CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN** 6](#_Toc200808799)

[**1.1** **Giới thiệu bài toán** 6](#_Toc200808800)

[**1.1.1** **Bài toán AI là gì** 6](#_Toc200808801)

[**1.1.2** **Input của bài toán** [1] 6](#_Toc200808802)

[**1.1.3** **Output của bài toán** 6](#_Toc200808803)

[**1.2** **Phạm vi thực hiện và tiềm năng của dự án** 6](#_Toc200808804)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 8](#_Toc200808805)

[**2.1** **Giới thiệu về thị giác máy tính** 8](#_Toc200808806)

[**2.1.1** **Khái niệm về thị giác máy tính** 8](#_Toc200808807)

[**2.1.2** **Ứng dụng thị giác máy tính** 8](#_Toc200808808)

[**2.2** **Giới thiệu về thuật toán DenseNet121** 9](#_Toc200808809)

[**2.3** **Giới thiệu về thuậtt toán Mobilenet** 10](#_Toc200808810)

[**2.4** **Giới thiệu về thuật toán VGG16** 12](#_Toc200808811)

[**CHƯƠNG 3: CƠ SỞ DỮ LIỆU** 14](#_Toc200808812)

[**3.1** **Chuẩn bị dữ liệu** 14](#_Toc200808813)

[**3.2** **Tiền xử lý dữ liệu** 15](#_Toc200808814)

[**CHƯƠNG 4: CƠ SỞ THỰC NGHIỆM** 24](#_Toc200808815)

[**4.1** **Thử nghiệm mô hình VGG16** 24](#_Toc200808816)

[**4.1.1** **Cài đặt mô hình** 24](#_Toc200808817)

[**4.1.2** **Huấn luyện mô hình** 24](#_Toc200808818)

[**4.2** **Thử nghiệm mô hình MobileNetV3Small** 26](#_Toc200808819)

[**4.2.1** **Cài đặt mô hình** 26](#_Toc200808820)

[**4.2.2** **Huấn luyện mô hình** 27](#_Toc200808821)

[**4.3** **Thử nghiệm mô ình DenseNet121** 28](#_Toc200808822)

[**4.3.1** **Cài đặt mô hình** 28](#_Toc200808823)

[**4.3.2** **Huấn luyện mô hình** 29](#_Toc200808824)

[**4.4** **Đánh giá hiệu suất và so sánh** 30](#_Toc200808825)

[**4.4.1** **So sánh Accuracy và Loss** 31](#_Toc200808826)

[**4.4.2** **So sánh Accuracy, Precision, Recall, F1-score** 33](#_Toc200808827)

[**4.5** **Đề xuất cải tiến** 35](#_Toc200808828)

[**4.5.1** **Tăng cường dữ liệu (Data Augmentation)** 35](#_Toc200808829)

[**4.5.2** **Thêm Early Stopping và Model Checkpoint** 37](#_Toc200808830)

[**4.6** **Dự đoán hình ảnh bằng Streamlet với mô hình MobileNet** 38](#_Toc200808831)

[**KẾT LUẬN** 42](#_Toc200808832)

[**TÀI LIỆU THAM KHẢO** 44](#_Toc200808833)

**DANH MỤC HÌNH ẢNH**

[Hình 1: Kiến trúc mạng học sâu DenseNet121. 14](#_Toc200807803)

[Hình 2: Mô tả Kiến trúc của DenseNet trong dự đoán hình ảnh. 14](#_Toc200807804)

[Hình 3: Kiến thức mạng nơ-ron phân tách theo chiều sâu của MobileNet 15](#_Toc200807805)

[Hình 4: Kiến trúc mạng MobileNet [7] 16](#_Toc200807806)

[Hình 5: Kiến trúc mô hình VGG16 17](#_Toc200807807)

[Hình 6: : Tập dữ liệu hình ảnh con vật 19](#_Toc200807808)

[Hình 7: Tải dữ liệu lên Google cololab 20](#_Toc200807809)

[Hình 8: Giải nén file dữ liệu zip 20](#_Toc200807810)

[Hình 9: Khởi tạo mô hình cần thiết 21](#_Toc200807811)

[Hình 10: Tổ chức cấu trúc dữ liệu mục tiêu 22](#_Toc200807812)

[Hình 11: Thiết lập thông tin chỉ số hình ảnh 23](#_Toc200807813)

[Hình 12: Chia dữ liệu thành từng phần để  tập huấn 24](#_Toc200807814)

[Hình 13: Hiển thị danh sách chứa cách lớp 24](#_Toc200807815)

[Hình 14: Chuẩn hóa dữ liệu hình ảnh trong các tập 25](#_Toc200807816)

[Hình 15: Phân Phối lớp 26](#_Toc200807817)

[Hình 16: Tập dữ liệu kiểm tra mỗi lớp 27](#_Toc200807818)

[Hình 17: Ảnh phân loại các lớp 28](#_Toc200807819)

[Hình 18: Cài đặt mô hình VGG16 29](#_Toc200807820)

[Hình 19: Huấn luyện mô hình VGG16 30](#_Toc200807821)

[Hình 20: Quá trình huấn luyện VGG16 31](#_Toc200807822)

[Hình 21: Cài đặt mô hình MobileNetV3Small 31](#_Toc200807823)

[Hình 22: Huấn luyện mô hình MobileNetV3Small 33](#_Toc200807824)

[Hình 23: Quá trình huấn luyên mô hình **MobileNetV3** 33](#_Toc200807825)

[Hình 24: Cài đặt mô hình DenseNet121 34](#_Toc200807826)

[Hình 25: Huấn luyện mô hình 35](#_Toc200807827)

[Hình 26: Biểu đồ so sánh độ chính xác trong huấn luyện và kiểm tra 36](#_Toc200807828)

[Hình 27: Biểu đồ so sánh độ mất mát trong huấn luyện và kiểm tra. 37](#_Toc200807829)

[Hình 28: Biểu đồ so sánh các chỉ số độ chính xác, độ chính xác dự đoán, độ nhạy và F1-score của các mô hình. 39](#_Toc200807830)

[Hình 29: Tăng cường dữ liệu 40](#_Toc200807831)

[Hình 30: Code tăng cường dữ liệu 1 41](#_Toc200807832)

[Hình 31: Code tăng cường dữ liệu 2 41](#_Toc200807833)

[Hình 32: Code kĩ thuật  Early Stopping và Model Checkpoint 42](#_Toc200807834)

[Hình 33: Code ứng dụng nhận diện động vật 1 44](#_Toc200807835)

[Hình 34: Code ứng dụng nhận diện động vật 2 45](#_Toc200807836)

[Hình 35: Kết quả dự đoán hình ảnh bằng Streamlet với mô hình MobileNet 46](#_Toc200807837)

# **DANH MỤC BẢNG BIỂU**

[Bảng 1: Kết luận so sánh hiệu suất các mô hình học sâu trong huấn luyện và kiểm tra. 32](#_Toc200791696)

[Bảng 2: So sánh các chỉ số hiệu suất của các mô hình học sâu. 33](#_Toc200791697)

**CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN**

1. **Giới thiệu bài toán**
2. **Bài toán AI là gì**
3. **Input của bài toán** [1]

Dữ liệu đầu vào của bài toán là các hình ảnh chứa một hoặc nhiều loài động vật. Các ảnh này có thể có nguồn gốc từ:

* Bộ dữ liệu huấn luyện được thu thập từ trước, ví dụ như tập dữ liệu "Animal Data" trên Kaggle.
* Hình ảnh từ camera thực tế trong môi trường tự nhiên hoặc tại sở thú.
* Hình ảnh do người dùng tải lên hệ thống.

Các hình ảnh đầu vào có thể có nhiều điều kiện khác nhau về độ phân giải, góc chụp, ánh sáng, phông nền… Điều này đòi hỏi mô hình cần có khả năng tổng quát hóa tốt và xử lý đa dạng các loại hình ảnh.

1. **Output của bài toán**

Kết quả đầu ra là tên của loài động vật có trong hình ảnh đầu vào. Với mỗi hình ảnh, hệ thống sẽ thực hiện các bước xử lý để nhận diện đối tượng động vật, trích xuất đặc trưng và so sánh với các lớp đã được huấn luyện để đưa ra dự đoán chính xác.

Nếu ảnh đầu vào chứa nhiều loài động vật, hệ thống cần có khả năng phân đoạn từng đối tượng (object detection) trước khi tiến hành nhận dạng từng cá thể. Mỗi kết quả nhận dạng sẽ đi kèm với một chỉ số độ tin cậy (confidence score) nhằm đánh giá xác suất dự đoán đúng của mô hình.

Ví dụ:

* Input: Ảnh một con voi.
* Output: "Voi" (Confidence: 96%).
* Input: Ảnh có một con mèo và một con chó.
* Output: "Mèo" (91%), "Chó" (93%).

1. **Phạm vi thực hiện và tiềm năng của dự án**

Để triển khai bài toán này, nhóm sử dụng tập dữ liệu "Animal Data" từ nền tảng Kaggle do người dùng likhon148 đăng tải. Tập dữ liệu bao gồm ảnh của 15 loài động vật khác nhau, được tổ chức thành các thư mục theo tên loài và đã bao gồm ảnh gốc lẫn ảnh được tăng cường (augmented). Việc sử dụng tập dữ liệu này rất thuận lợi cho việc huấn luyện mô hình học có giám sát.

Các bước thực hiện chính bao gồm:

* Tiền xử lý dữ liệu: chuẩn hóa kích thước ảnh, cân bằng số lượng ảnh giữa các lớp.
* Huấn luyện mô hình CNN từ đầu hoặc sử dụng mô hình pretrained (MobileNet, ResNet, EfficientNet...).
* Thực hiện đánh giá mô hình bằng các chỉ số như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (precision).
* Xây dựng giao diện đơn giản (web hoặc app) cho phép người dùng tải ảnh và nhận kết quả nhận diện. [2]

Tiềm năng ứng dụng của hệ thống nhận diện động vật bằng AI là rất rộng mở, có thể kể đến như:

* Trong giáo dục: hỗ trợ học sinh, sinh viên học nhận biết động vật một cách trực quan và sinh động.
* Trong bảo tồn thiên nhiên: nhận diện loài từ ảnh chụp tự động trong rừng giúp theo dõi sự xuất hiện của các loài quý hiếm.
* Trong nông nghiệp và chăn nuôi: giám sát vật nuôi tự động qua camera.
* Trong du lịch – giải trí: tích hợp vào ứng dụng nhận diện động vật tại sở thú, công viên, bảo tàng.
* Phát triển thêm: tích hợp xử lý video để theo dõi động vật theo thời gian thực, nhận dạng hành vi, cảnh báo động vật nguy hiểm.

Từ một mô hình cơ bản nhận diện ảnh tĩnh, hệ thống hoàn toàn có thể phát triển thành công cụ hỗ trợ đa ngành với độ chính xác ngày càng cao nhờ vào lượng dữ liệu lớn và sự phát triển của công nghệ học sâu.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

1. **Giới thiệu về thị giác máy tính**
2. **Khái niệm về thị giác máy tính**

Thị giác máy tính là một lĩnh vực của trí tuệ nhân tạo và khoa học máy tính, tập trung vào việc xây dựng các hệ thống có khả năng nhận biết, phân tích và hiểu hình ảnh cũng như video tương tự như con người. Với sự phát triển mạnh mẽ của công nghệ, các hệ thống thị giác máy tính ngày nay có thể truy cập và xử lý khối lượng lớn dữ liệu hình ảnh và video từ nhiều nguồn khác nhau như điện thoại thông minh, camera giao thông, hệ thống an ninh và các thiết bị IoT. Ứng dụng của công nghệ này bao gồm nhận diện đối tượng, nhận diện khuôn mặt, phân loại hình ảnh, giám sát, phát hiện và đưa ra đề xuất thông minh. Nhờ vào sự kết hợp giữa trí tuệ nhân tạo và học máy (AI/ML), thị giác máy tính ngày càng trở nên chính xác và hiệu quả hơn trong việc xử lý dữ liệu thị giác phục vụ cho nhiều mục đích khác nhau trong đời sống và công nghiệp [3]. Quá trình mô phỏng thị giác con người này được chia thành 3 giai đoạn nối tiếp (tương tự cách con người nhìn): mô phỏng mắt (thu nhận - khó), mô phỏng vỏ não thị giác (xử lý - rất khó) và mô phỏng phần còn lại của bộ não (phân tích - khó nhất). [4]

Thị giác máy tính bao gồm các lĩnh vực chính sau:

* Xử lý hình ảnh: Phát triển các kỹ thuật xử lý ảnh như tăng/giảm chất lượng hình ảnh, loại bỏ nhiễu.
* Nhận mẫu giao diện: Ứng dụng các kỹ thuật đa dạng để phân tích và diễn giải các mẫu trong dữ liệu hình ảnh.
* Quang trắc: Liên quan đến công việc thu thập các số đo chính xác từ hình ảnh.

Các lĩnh vực này kết hợp với nhau tạo thành cơ sở vững chắc để phát triển các hệ thống thị giác máy tính, giúp con người hiểu rõ hơn và tương tác hiệu quả hơn với thế giới xung quanh thông qua dữ liệu hình ảnh và video. [5]

1. **Ứng dụng thị giác máy tính**

Thị giác máy tính được ứng dụng rộng rãi trong thực tiễn, với vai trò quan trọng trong nhiều lĩnh vực như kinh doanh, giải trí, giao thông, y tế và cả trong đời sống thường nhật. Các ứng dụng cụ thể của thị giác máy tính:

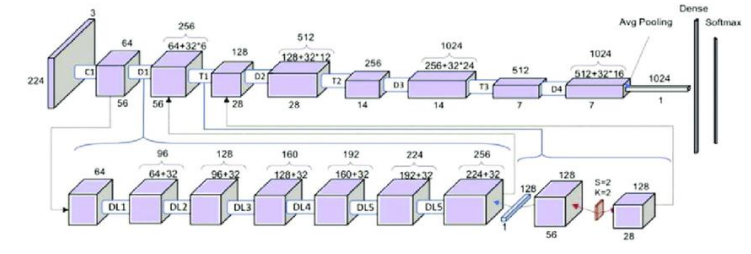
* Phân loại hình ảnh: Phân loại hình ảnh giúp máy tính nhận biết và xác định chính xác một hình ảnh thuộc về loại nào. Thị giác máy tính hiểu rõ và gắn nhãn tương ứng, ví dụ như cây cối, máy bay hoặc tòa nhà. Ví dụ như camera có thể nhận diện khuôn mặt trong ảnh và lấy nét khuôn mặt.
* Nhận diện vật thể: Phát hiện đối tượng là một tác vụ trong thị giác máy tính nhằm nhận diện và xác định vị trí các đối tượng trong hình ảnh hoặc video. Công nghệ này được ứng dụng trong tự động hóa, giám sát sản xuất và camera thông minh để phát hiện người hoặc vật thể trong thời gian thực và đưa ra cảnh báo cho người dùng.
* Theo dõi đối tượng: Theo dõi đối tượng sử dụng mô hình học sâu để nhận diện và theo dõi các đối tượng qua nhiều khung hình. Quá trình này bắt đầu bằng việc phát hiện đối tượng, tạo ô viền, gán ID và theo dõi liên tục. Công nghệ này được ứng dụng trong giám sát giao thông, theo dõi con người và chẩn đoán hình ảnh y khoa.
* Phân đoạn: Phân đoạn là một thuật toán trong thị giác máy tính dùng để chia hình ảnh thành các vùng dựa trên điểm ảnh, giúp xác định rõ ranh giới và hình dạng của đối tượng. Kỹ thuật này giúp đơn giản hóa hình ảnh và phát hiện nếu có nhiều đối tượng xuất hiện trong cùng một khung hình.
* Truy xuất hình ảnh dựa trên nội dung: Truy xuất hình ảnh dựa trên nội dung cho phép tìm kiếm hình ảnh kỹ thuật số trong cơ sở dữ liệu lớn. Hệ thống phân tích siêu dữ liệu như thẻ, mô tả, nhãn và từ khóa. [3]

Những ứng dụng trên cho thấy khả năng mạnh mẽ của thị giác máy tính trong việc nâng cao hiệu quả và thúc đẩy tự động hóa trên nhiều lĩnh vực trong đời sống và sản xuất công nghiệp.

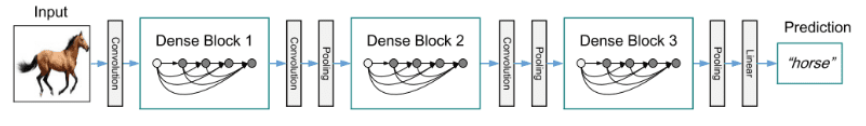
1. **Giới thiệu về thuật toán DenseNet121**

DenseNet121 là một biến thể của kiến trúc DenseNet do Gao Huang và cộng sự phát triển, được đánh giá là một bước tiến quan trọng trong thị giác máy tính. Mô hình này khắc phục các hạn chế của CNN truyền thống bằng cách sử dụng kết nối dày đặc (dense connectivity), trong đó mỗi lớp nhận đầu vào từ tất cả các lớp trước đó trong cùng khối. Nhờ đó, DenseNet121 tăng cường khả năng truyền gradient, tái sử dụng đặc trưng hiệu quả và giảm số lượng tham số, từ đó nâng cao hiệu suất học và giảm thiểu chi phí tính toán.

Kiến trúc của DenseNet121 bao gồm các thành phần chính: các Dense Block và Transition Layer. Mỗi Dense Block chứa nhiều lớp tích chập 3x3, với đặc trưng được nối theo chiều kênh từ các lớp trước (channel-wise concatenation), còn Transition Layer thực hiện giảm số lượng kênh và kích thước bản đồ đặc trưng qua tích chập 1x1 và gộp trung bình 2x2. Mô hình bắt đầu bằng tích chập 7x7 và max pooling 3x3, sau đó là 4 Dense Block với số lớp lần lượt là 6, 12, 24 và 16, kết thúc bằng Global Average Pooling và lớp Fully Connected để phân loại. [6]



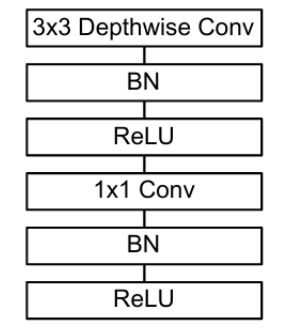
Hình 1: Kiến trúc mạng học sâu DenseNet121.



Hình 2: Mô tả Kiến trúc của DenseNet trong dự đoán hình ảnh.

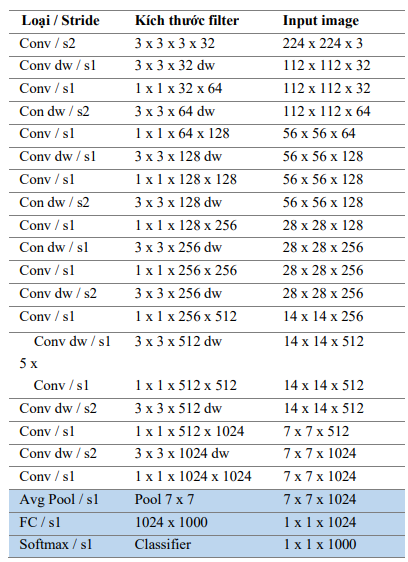
1. **Giới thiệu về thuậtt toán Mobilenet**

MobileNet là một mô hình do nhóm nghiên cứu của Google giới thiệu vào năm 2017 trong bài báo “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. Mô hình này được thiết kế nhỏ gọn, độ trễ thấp, phù hợp với các ứng dụng thị giác trên thiết bị nhúng và di động. MobileNet sử dụng kiến trúc mạng nơ-ron tích chập phân tách theo chiều sâu (depthwise separable convolutions), ngoại trừ lớp đầu tiên là tích chập thông thường. Tổng thể mô hình gồm 28 lớp [7]. MobileNet được xây dựng dựa trên nền tảng một dạng phức hợp cho phép biến đổi phép tích chập thông thường thành một phép tích chập theo chiều sâu (Depthwise Convolution) và một phép tích chập theo từng điểm (Pointwise Convolution). Mạng MobileNet có nhiều thế hệ như MobileNet-v1, MobileNet-v2, MobileNet-v3, trong đó, mạng MobileNet-v1 được sử dụng để làm mạng cơ sở thực thi tốt hơn trên các bo nhúng có kích thước nhỏ [8].



Hình 3: Kiến thức mạng nơ-ron phân tách theo chiều sâu của MobileNet

Kiến trúc mạng nơ-ron tích chập phân tách theo chiều sâu của MobileNet được xây dựng từ một lớp tích chập theo chiều sâu kích thước 3×3, dùng để trích xuất các đặc trưng cơ bản từ hình ảnh. Lớp tích chập này áp dụng các bộ lọc riêng lẻ cho từng kênh đầu vào, giúp giảm đáng kể số lượng tham số so với tích chập truyền thống. Sau đó, mạng sử dụng các lớp tích chập điểm (1×1) để kết hợp các đặc trưng thu được từ các kênh đầu ra của lớp tích chập theo chiều sâu. Kiến trúc các tầng của MobileNet được minh họa trong hình dưới đây.



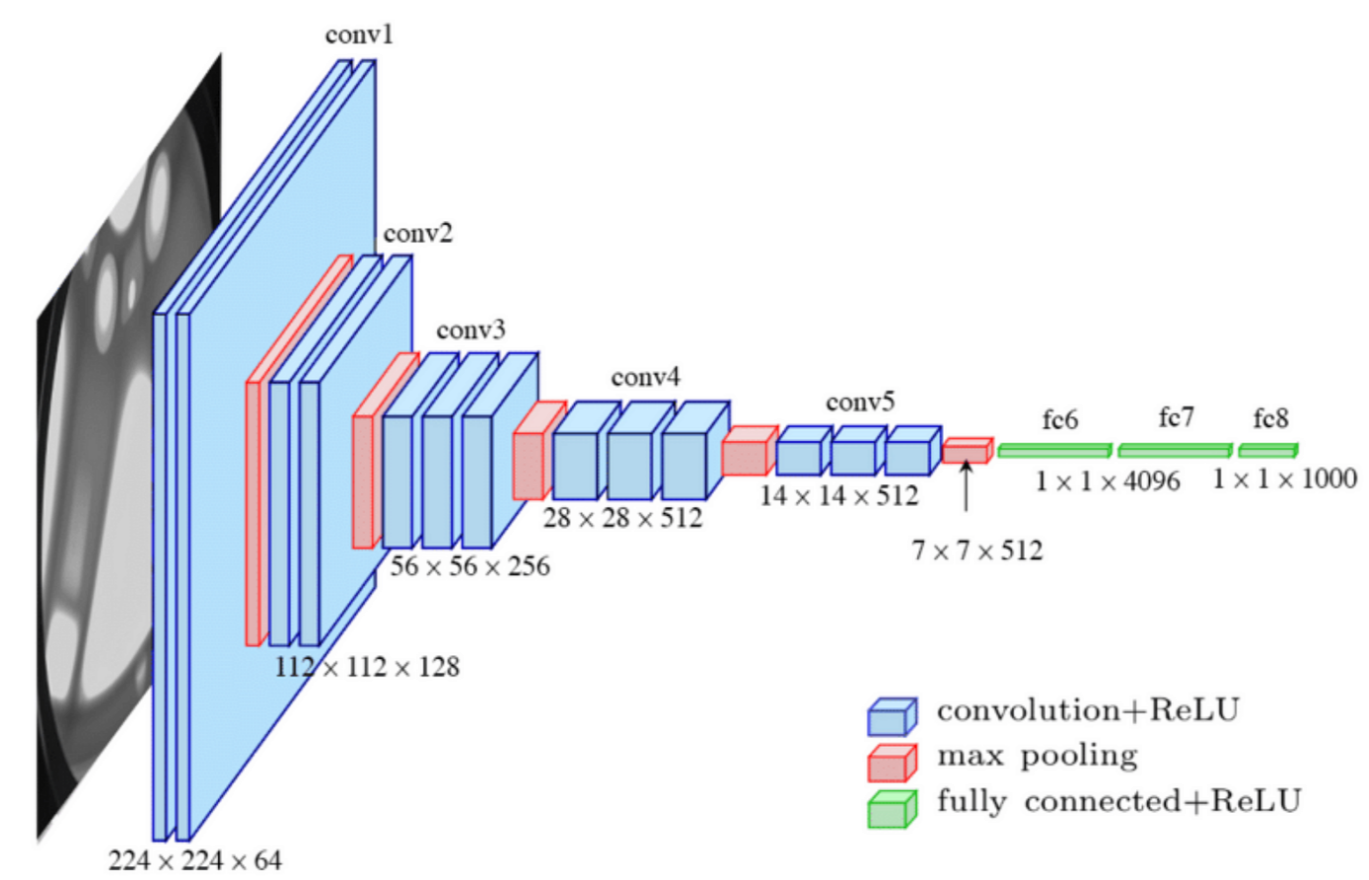
Hình 4: Kiến trúc mạng MobileNet [7]

Mạng MobileNet gồm 28 lớp, được tổ chức thành 13 khối lặp lại. Mỗi khối bao gồm một lớp tích chập phân tách theo chiều sâu, kết hợp giữa lớp tích chập theo chiều sâu và lớp tích chập điểm. Ngoài các khối này, kiến trúc còn có lớp tích chập ban đầu, lớp tổng hợp trung bình, lớp kết nối đầy đủ và lớp softmax, giúp tính toán xác suất hình ảnh thuộc về từng lớp phân loại.

1. **Giới thiệu về thuật toán VGG16**

VGG16 là một kiến trúc mạng nơ-ron tích chập (CNN) nổi tiếng, được phát triển bởi nhóm Visual Geometry Group (VGG) của Đại học Oxford và công bố trong bài báo “Very Deep Convolutional Networks for Large-Scale Image Recognition” vào năm 2014. Mô hình này đã đạt kết quả xuất sắc tại cuộc thi ImageNet Challenge (ILSVRC) 2014, nổi bật nhờ vào kiến trúc đơn giản nhưng hiệu quả cao. Với tổng cộng 16 lớp có trọng số học được, VGG16 nhanh chóng trở thành một trong những mô hình nền tảng được sử dụng rộng rãi trong các ứng dụng thị giác máy tính. [9]

Về mặt kiến trúc, VGG16 dựa trên mô hình CNN sâu sử dụng các lớp convolution 3×3 với stride = 1 và padding “same”, đi kèm với các lớp max‑pooling 2×2 để giảm kích thước tín hiệu. Các khối convolution được chồng lên nhau theo cấu trúc đơn giản nhưng rất hiệu quả: từ 64 filter, tăng dần qua nhiều tầng đến 512 filter, giúp mô hình học các đặc trưng phức tạp. Cuối cùng là ba lớp fully connected với hai lớp đầu có 4096 neuron, và lớp softmax cuối cùng để phân loại. Tổng cộng mô hình có khoảng 138 triệu tham số, tiêu biểu cho sự cải thiện đáng kể so với các kiến trúc trước đó dù giữ được thiết kế tuần tự và dễ hiểu [10]



Hình 5: Kiến trúc mô hình VGG16

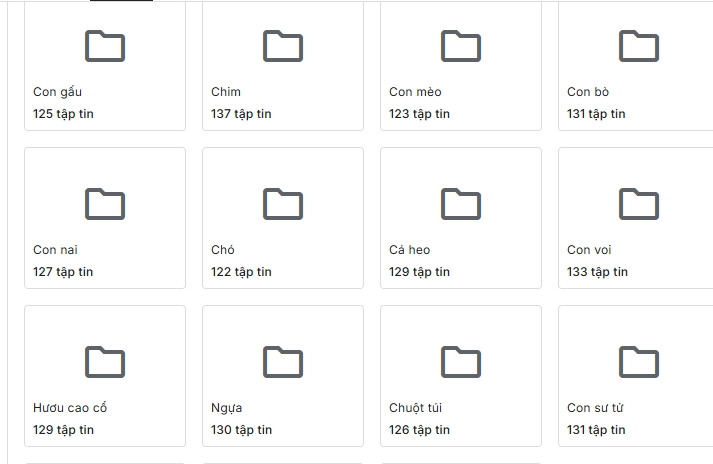
VGG16 được sử dụng phổ biến trong các tác vụ như phân loại ảnh, trích xuất đặc trưng, và học chuyển giao (transfer learning). Trong thực tế, mô hình thường được sử dụng ở dạng pretrained trên ImageNet, sau đó tinh chỉnh (fine-tune) để áp dụng cho các bài toán cụ thể như nhận diện khuôn mặt, phân loại y tế, hay nông nghiệp thông minh. [11]

**CHƯƠNG 3: CƠ SỞ DỮ LIỆU**

1. **Chuẩn bị dữ liệu**

Dữ liệu đầu vào đóng vai trò vô cùng quan trọng trong quá trình xây dựng và huấn luyện mô hình học sâu. Một tập dữ liệu chất lượng, được chuẩn bị cẩn thận sẽ giúp mô hình học hiệu quả hơn, giảm thời gian huấn luyện và tăng độ chính xác. Nhóm chúng em đã lấy dữ liệu sẵn có từ trên Kaggle. Tập dữ liệu bao gồm 1944 hình ảnh con vật và được phân chia thành từng nhãn dán khác nhau bao gồm:

* 137 hình ảnh con ngựa vằn
* 129 hình ảnh con hổ
* 135 hình ảnh con gấu trúc
* 131 hình ảnh con sư tử
* 126 hình ảnh con Kangaroo
* 130 hình ảnh con ngựa
* 129 hình ảnh con hươu cao cổ
* 133 hình ảnh con voi
* 129 hình ảnh con cá heo
* 122 hình ảnh con chó
* 127 hình ảnh con nai
* 131 hình ảnh con bò
* 123 hình ảnh con mèo
* 137 hình ảnh con chim
* 125 hình ảnh con gấu



Hình 6: : Tập dữ liệu hình ảnh con vật

Việc tổ chức dữ liệu này đảm bảo rằng mỗi loại con vật  có đủ số lượng ảnh để huấn

luyện và kiểm tra mô hình một cách hiệu quả, giúp tăng độ chính xác và khả năng tổng quát hóa của mô hình trong thực tế.

1. **Tiền xử lý dữ liệu**

Nhận diện hình ảnh động vật là một bài toán yêu cầu mô hình học sâu phải xử lý các dữ liệu hình ảnh đa dạng, phức tạp, có thể bị nhiễu, không đồng nhất về kích thước, màu sắc hoặc góc chụp. Do đó, tiền xử lý dữ liệu là một bước vô cùng quan trọng nhằm đảm bảo chất lượng ảnh đầu vào, đồng thời giúp mô hình học tốt hơn và đưa ra kết quả chính xác hơn.

Quá trình tiền xử lý sẽ chuẩn hoá ảnh, đồng bộ kích thước và chuyển đổi không gian màu phù hợp với yêu cầu của các mạng học sâu như DenseNet121, MobileNet và VGG16 . Việc chuẩn hóa ảnh cũng giúp giảm tải tính toán, đồng thời làm nổi bật các đặc trưng hình ảnh cần thiết cho việc huấn luyện mô hình hiệu quả.

Mục tiêu của bước tiền xử lý dữ liệu

* Tự động tải toàn bộ ảnh từ thư mục dữ liệu được tổ chức theo từng lớp tương ứng với tên loài động vật.
* Chuyển đổi ảnh từ định dạng gốc sang không gian màu RGB chuẩn đầu vào của các mô hình học sâu.
* Thay đổi kích thước toàn bộ ảnh về một kích thước cố định (150x150 pixels), phù hợp với yêu cầu đầu vào của các kiến trúc mạng nêu trên.
* Chuẩn hóa các giá trị điểm ảnh (pixel) về khoảng [0, 1] nhằm tăng tốc độ hội tụ của quá trình huấn luyện.
* Lưu trữ ảnh và nhãn dưới dạng mảng NumPy để sẵn sàng sử dụng trong quá trình huấn luyện mô hình.

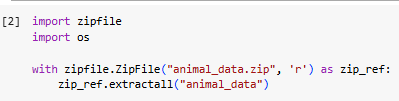
**Bước 1**:  Tải file dữ liệu lên môi trường Google Colab.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXeIHlIHManDrZOaD4yqBQWHXwxKRizFpQ2goyExQZ6HVwWMmU9VzwzOCpUsqIxojktl1DcJW8U5O9cTdWfAWrouKDzncRgtlo5lc2QltmkTGpGAt4kFFlGoK1DnQJj_yhm8Lj0npg?key=QL0ertfNaQCelRQ2ih0KOA

Hình 7: Tải dữ liệu lên Google cololab

Sử dụng lệnh file.upload()  để tải tập tin dữ liệu từ máy tính cá nhân lên Google Colab. Đây là bước đầu tiên để chuẩn bị dữ liệu cho quá trình tiền xử lý và huấn luyện mô hình.

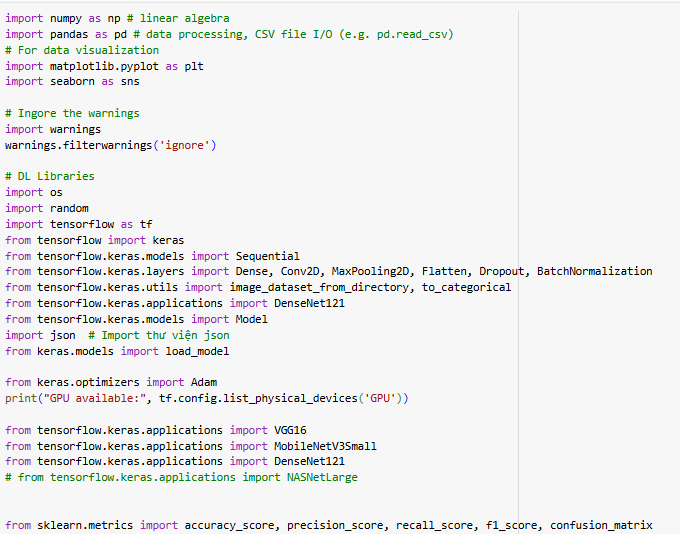
**Bước 2**: Giải nén file dữ liệu zip



Hình 8: Giải nén file dữ liệu zip

Tập tin animal\_data.zip sau khi được tải lên sẽ được giải nén vào thư mục animal\_data. Việc này giúp truy cập trực tiếp vào các ảnh phục vụ cho bước tiền xử lý và huấn luyện mô hình.

**Bước 3:** Khởi tạo và nhập mô hình cần thiết



Hình 9: Khởi tạo mô hình cần thiết

Các thư viện cần thiết bao gồm:

* Thư viện numpy: Dùng cho tính toán số học, đại số tuyến tính. Mục đích: Xử lý mảng đa chiều, toán học trong AI/ML.
* Thư viện pandas: Xử lý dữ liệu dạng bảng. Mục đích: Đọc/ghi file CSV, xử lý dữ liệu structured (DataFrame)
* Module pyplot từ matplotlib (vẽ đồ thị). Mục đích: Hiển thị hình ảnh, đồ thị
* Thư viện seaborn: Trực quan hóa dữ liệu thống kê. Mục đích: Vẽ biểu đồ đẹp hơn so với matplotlib.
* Thư viện os: Hỗ trợ thao tác với hệ thống tệp.
* Thư viện random để tạo số ngẫu nhiên.
* Thư viện tensorflow: Xây dựng, huấn luyện mô hình neural network.
* Kiến trúc VGG16 - một mô hình CNN kinh điển với 16 lớp, được huấn luyện trên ImageNet
* Kiến trúc MobileNetV3Small - phiên bản nhẹ, tối ưu cho thiết bị di động.
* Kiến trúc DenseNet121 - mô hình sử dụng kết nối dày đặc (dense connections) giữa các lớp. Mục đích: Giảm vanishing gradient, phù hợp với dataset nhỏ.

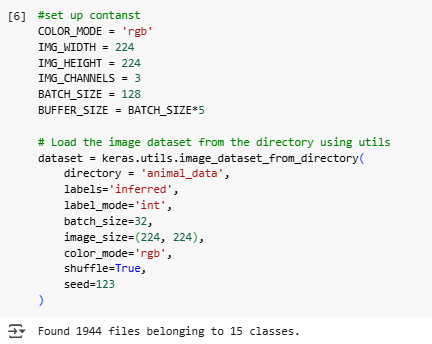
**Bước 4**: Tổ chức cấu trúc thư mục dữ liệu



Hình 10: Tổ chức cấu trúc dữ liệu mục tiêu

Do một số dữ liệu sau khi giải nén có thể bị lồng nhiều cấp thư mục (ví dụ: animal\_data/animal\_data/...), nhóm sử dụng thư viện shutil để di chuyển toàn bộ nội dung thư mục con về đúng cấp thư mục chính

**Bước 5**: Thiết lập thông số xử lý hình ảnh



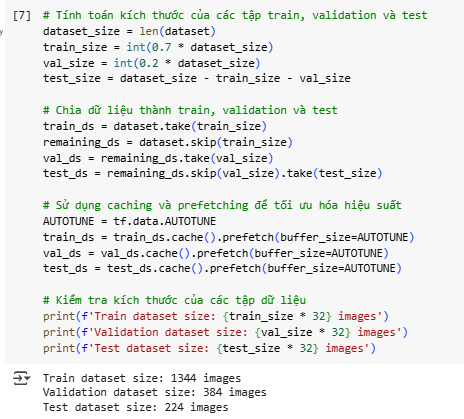
Hình 11: Thiết lập thông tin chỉ số hình ảnh

Khi mà tải dữ liệu về thì có những dữ liệu khồng đồng nhất với nhau và ảnh ở hình chữ nhật rất tốn dung lượng, Cài đặt đặt lại tham số để đưa hình ảnh về dạng hình vuông, định dạng màu của ảnh là rgb

* Directory='animal\_data': Đường dẫn đến thư mục chứa ảnh (đã được sửa cấu trúc ở bước trước).
* Labels='inferred': Nhãn tự động suy ra từ tên thư mục con (ví dụ: animal\_data/cat/ → nhãn cat).
* Label\_mode='int': Chuyển nhãn thành số nguyên
* Image\_size=(224, 224): Resize ảnh về kích thước 224x224.
* Color\_mode='rgb': Đọc ảnh với 3 kênh màu.
* Shuffle=True: Xáo trộn dữ liệu để tránh bias.
* Seed=123: Cố định random seed để tái lặp kết quả.

Số lượng ảnh nhận được là 1944 ảnh, thuộc 15 lớp (classes)

**Bước 6**: Chia dữ liệu thành tập huấn luyện, validation và kiểm tra



Hình 12: Chia dữ liệu thành từng phần để  tập huấn

Việc chia dữ liệu được thực hiện với độ chính xác cao bằng cách dùng phương thức skip() và take() trên dataset.

Dữ liệu được chia theo tỷ lệ:

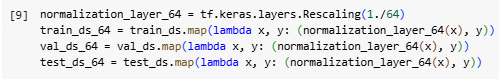
* 70% cho tập huấn luyện: 1334 ảnh
* 20% cho tập validation: 384 ảnh
* 10% cho tập kiểm tra: 224 ảnh



Hình 13: Hiển thị danh sách chứa cách lớp

Để phục vụ huấn luyện mô hình phân loại đa lớp, các nhãn được chuyển đổi sang định dạng one-hot encoding. Nhóm định nghĩa một hàm preprocess\_dataset() để thực hiện việc mã hóa này và áp dụng cho cả ba tập dữ liệu.

* Num\_classes = len(classes): Tính số lượng lớp (ở đây là 15).
* one-hot encoding cho cả 3 tập dữ liệu (train/val/test).



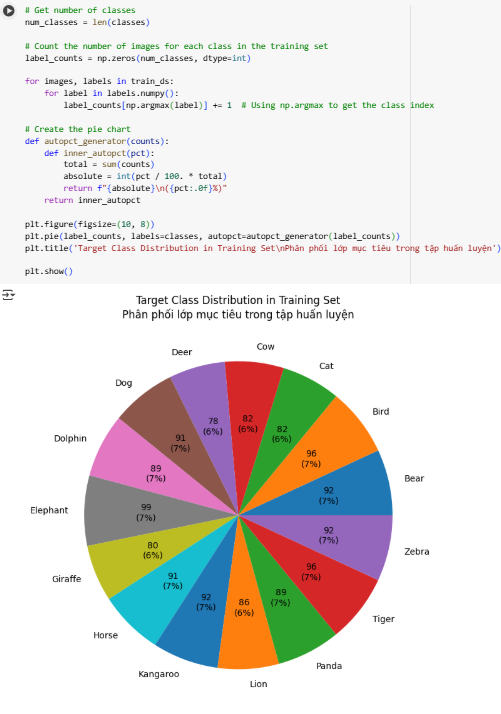
Hình 14: Chuẩn hóa dữ liệu hình ảnh trong các tập

Sau khi các ảnh được chia thành các tập huấn luyện, kiểm tra và validation, một bước tiền xử lý quan trọng tiếp theo là chuẩn hóa giá trị các pixel ảnh.

Việc chuẩn hóa này giúp:

* Tăng tốc độ hội tụ khi huấn luyện mô hình.
* Giảm độ lệch chuẩn giữa các đặc trưng đầu vào.
* Tránh hiện tượng gradient biến mất hoặc nổ trong các mạng sâu.

**Bước 7**: Hiển thị dữ liệu đầu vào



Hình 15: Phân Phối lớp

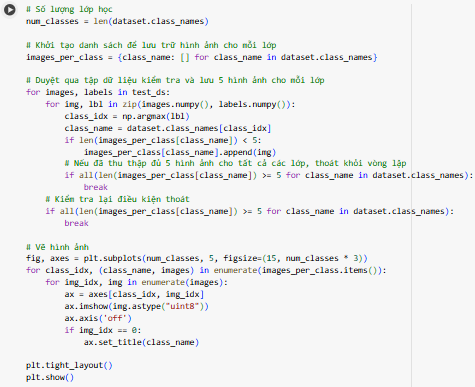
Các lệnh này thực hiện việc đếm số lượng mẫu cho mỗi lớp trong tập dữ liệu huấn luyện và sau đó trực quan hóa sự phân bố đó bằng biểu đồ tròn, giúp người dùng dễ dàng nhận biết sự cân bằng hoặc mất cân bằng giữa các lớp. Đây là một bước quan trọng trong quá trình tiền xử lý dữ liệu cho các bài toán phân loại trong học máy.

Dưới đây là tóm gọn các mục chính trong đoạn mã và biểu đồ:

1. **Đếm số lượng hình ảnh mỗi lớp:** Mã Python dùng để duyệt qua tập dữ liệu huấn luyện, đếm số lượng hình ảnh cho từng loại (lớp) đối tượng khác nhau.

2. **Tạo nhãn phần trăm tự động:** Một hàm được định nghĩa để tính toán và định dạng số lượng tuyệt đối cùng với phần trăm của mỗi lớp, hiển thị trực tiếp trên biểu đồ.

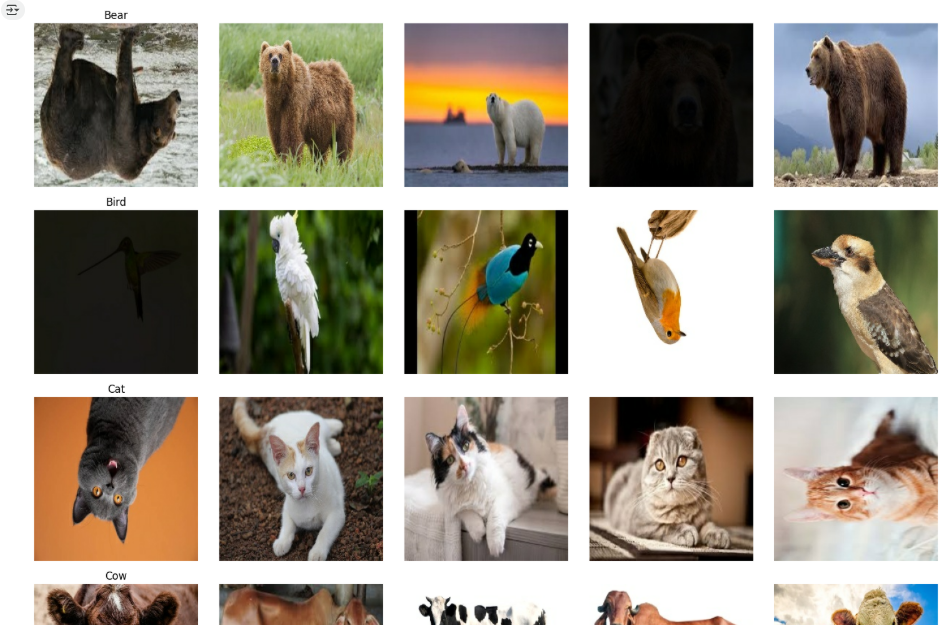
3. **Vẽ biểu đồ tròn phân phối lớp:** Sử dụng thư viện Matplotlib để tạo biểu đồ tròn, thể hiện tỷ lệ phần trăm và số lượng hình ảnh của mỗi lớp trong tập huấn luyện.



Hình 16: Tập dữ liệu kiểm tra mỗi lớp

* Khởi tạo: Thiết lập số lượng lớp và cấu trúc dữ liệu (từ điển) để lưu trữ hình ảnh được chọn.
* Thu thập hình ảnh: Duyệt qua tập dữ liệu kiểm tra, chọn ra tối đa 5 hình ảnh cho mỗi lớp và lưu trữ chúng. Quá trình dừng lại khi đã thu thập đủ 5 hình ảnh cho tất cả các lớp.
* Hiển thị: Tạo một lưới các ô con và vẽ 5 hình ảnh đã chọn của mỗi lớp lên đó, kèm theo tên lớp, để dễ dàng xem xét và kiểm tra dữ liệu.

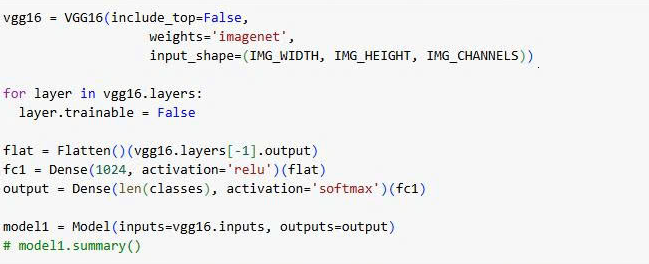
Kết quả



Hình 17: Ảnh phân loại các lớp

**CHƯƠNG 4: CƠ SỞ THỰC NGHIỆM**

1. **Thử nghiệm mô hình VGG16**
2. **Cài đặt mô hình**

****

Hình 18: Cài đặt mô hình VGG16

**include\_top=False**: Loại bỏ phần “đầu ra” (lớp Fully Connected cuối) của VGG16 gốc, cho phép ta tùy chỉnh phần phân loại theo số lớp mới.

**weights='imagenet'**: Sử dụng trọng số đã huấn luyện từ tập dữ liệu ImageNet giúp mô hình học đặc trưng tốt hơn từ ảnh

**input\_shape**: Xác định kích thước ảnh đầu vào (ví dụ: 224x224x3), đồng nhất với bước tiền xử lý hình ảnh đã chuẩn hóa trước đó.

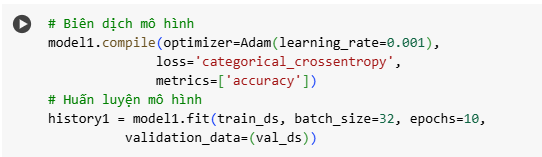
**Flatten()**: Chuyển đầu ra cuối cùng từ khối CNN của VGG16 (dạng tensor 3 chiều) thành vector 1 chiều.

**Dense(1024, activation='relu')**: Thêm một lớp ẩn với 1024 nơ-ron, sử dụng hàm kích hoạt **ReLU**.

**Dense(len(classes), activation='softmax')**: Lớp đầu ra cho mô hình với số nơ-ron bằng số lớp phân loại trong bài toán (ví dụ: 15 loài động vật).

**Model(inputs=vgg16.inputs, outputs=output)**: Tạo mô hình hoàn chỉnh kết hợp giữa phần thân của VGG16 và phần phân loại mới được thiết kế.

1. **Huấn luyện mô hình**

****

Hình 19: Huấn luyện mô hình VGG16

**optimizer=Adam(...)**: Sử dụng bộ tối ưu Adam – một thuật toán phổ biến trong deep learning, kết hợp giữa momentum và RMSProp, giúp mô hình hội tụ nhanh và ổn định.

**learning\_rate=0.001**: Tốc độ học (learning rate) đặt ở mức mặc định phổ biến, vừa đủ để mô hình học hiệu quả mà không bị dao động.

**loss='categorical\_crossentropy'**: Hàm mất mát dùng cho bài toán \*\*phân loại nhiều lớp\*\* với đầu ra là \*\*one-hot encoding\*\*.

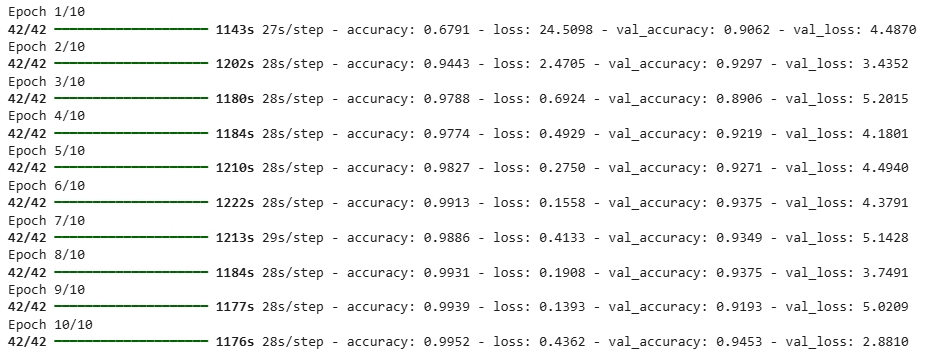
**metrics=['accuracy']**: Sử dụng độ chính xác làm chỉ số đánh giá trong quá trình huấn luyện và kiểm tra.

**train\_ds**: Tập dữ liệu huấn luyện đã được tiền xử lý và chuẩn hóa.

**batch\_size=32**: Chia dữ liệu thành từng lô gồm 32 ảnh để huấn luyện từng bước.

**epochs=10**: Lặp lại toàn bộ tập dữ liệu huấn luyện **10 lần**.

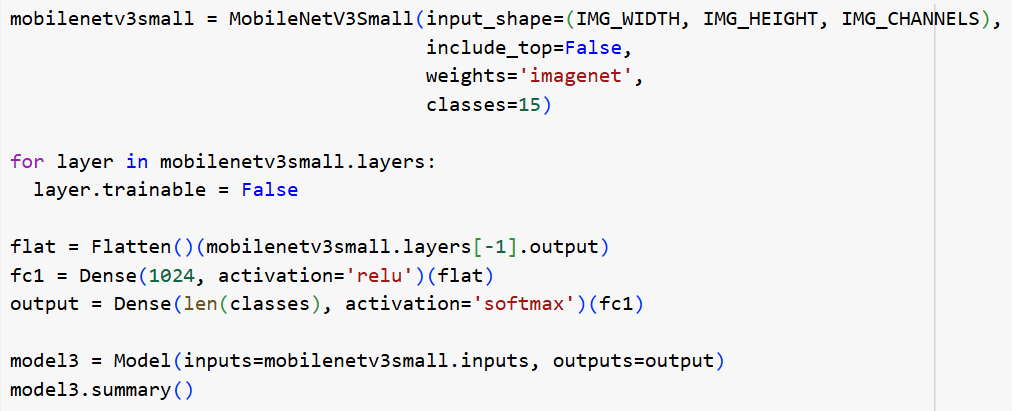
**validation\_data=val\_ds**: Tập kiểm tra dùng để đánh giá hiệu suất mô hình sau mỗi vòng huấn luyện (epoch).



Hình 20: Quá trình huấn luyện VGG16

Trong 10 vòng huấn luyện của mô hình VGG16, thời gian huấn luyện trung bình của mỗi vòng là khoảng 28 giây, tương đương chưa đến nửa phút. Độ lỗi thấp nhất đạt được trên tập đánh giá là 2.8810, và độ chính xác cao nhất trên tập đánh giá là 0.9453, tức là mô hình dự đoán đúng 94.53% số hình ảnh trong tập kiểm thử. Mô hình thể hiện khả năng học rất nhanh với độ chính xác huấn luyện đạt gần tuyệt đối từ epoch 3 trở đi, đồng thời giữ mức độ chính xác kiểm tra ổn định và cao qua các vòng huấn luyện. Tuy có một số dao động nhẹ ở val\_loss, nhưng nhìn chung mô hình VGG16 cho thấy hiệu suất tốt và khả năng tổng quát hóa khá tốt trên tập dữ liệu động vật.

1. **Thử nghiệm mô hình MobileNetV3Small**
2. **Cài đặt mô hình**



Hình 21: Cài đặt mô hình MobileNetV3Small

**input\_shape=(IMG\_WIDTH, IMG\_HEIGHT, IMG\_CHANNELS)**: Định nghĩa đầu vào của mô hình là ảnh màu có kích thước do người dùng định nghĩa (ví dụ: 224×224×3).

**include\_top=False:** Không sử dụng các lớp phân loại gốc (top layers) của MobileNetV3Small - điều này cho phép ta tự thiết kế phần phân loại (head) riêng.

**weights='imagenet':** Sử dụng trọng số đã được huấn luyện trước trên bộ dữ liệu **ImageNet**, giúp mô hình học được các đặc trưng chung của ảnh.

**classes=15:** Tham số này không ảnh hưởng trong trường hợp include\_top=False, nhưng thể hiện rằng bài toán có 15 lớp phân loại.

**Flatten():** Chuyển đổi đầu ra của mô hình nền (từ tensor nhiều chiều thành vector 1 chiều) để đưa vào các lớp fully connected.

**Dense(1024, activation='relu'):** Thêm một lớp Dense với 1024 nơ-ron, dùng hàm kích hoạt ReLU để học các mối quan hệ phi tuyến.

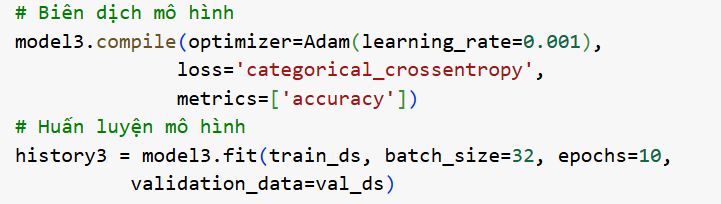
**Dense(len(classes), activation='softmax'):** Lớp phân loại cuối cùng với số nơ-ron bằng số lớp (len(classes)), dùng softmax để xuất xác suất cho từng lớp.

**inputs=mobilenetv3small.inputs:** Sử dụng đầu vào của MobileNetV3Small làm đầu vào của mô hình.

**outputs=output:** Sử dụng phần “head” (các lớp mới thêm) làm đầu ra cho mô hình.

**model3.summary()**: In ra kiến trúc mô hình.

1. **Huấn luyện mô hình**



Hình 22: Huấn luyện mô hình MobileNetV3Small

**optimizer=Adam(...):** Sử dụng thuật toán tối ưu Adam – là sự kết hợp giữa momentum và RMSProp, thường cho kết quả tốt và ổn định.  
 **learning\_rate=0.001:** Tốc độ học (learning rate) là 0.001, đây là giá trị mặc định phổ biến và khá ổn định.

**loss='categorical\_crossentropy':** Hàm mất mát dành cho bài toán phân loại đa lớp khi nhãn đầu ra ở dạng one-hot encoding.

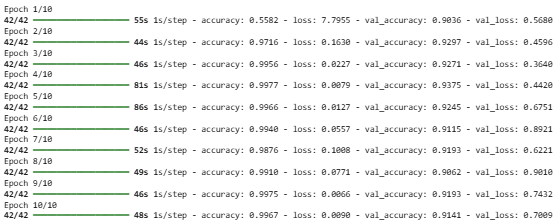
**metrics=['accuracy']:** Sử dụng độ chính xác (accuracy) làm chỉ số để đánh giá hiệu suất mô hình trong quá trình huấn luyện.

**train\_ds:** Tập dữ liệu huấn luyện, gồm các ảnh và nhãn tương ứng.

**batch\_size=32:** Mỗi lần huấn luyện (bước) sẽ sử dụng 32 ảnh (mẫu) một lúc để cập nhật trọng số.

**epochs=10:** Mô hình sẽ đi qua toàn bộ tập huấn luyện 10 lần.

**validation\_data=val\_ds:** Tập kiểm tra (validation set) được dùng để đánh giá mô hình sau mỗi epoch (vòng huấn luyện).



Hình 23: Quá trình huấn luyên mô hình **MobileNetV3**

Trong 10 vòng huấn luyện của mô hình MobileNetV3, thời gian huấn luyện trung bình của mỗi vòng là khoảng 58 giây, tương đương gần 01 phút. Độ lỗi thấp nhất đạt được trên tập đánh giá là 0.3648 và độ chính xác cao nhất trên tập đánh giá là 0.9375, có nghĩa là mô hình dự đoán đúng 93.75% số hình ảnh của tập kiểm thử. Mô hình cho thấy khả năng học tốt với độ chính xác ổn định qua các vòng huấn luyện.

1. **Thử nghiệm mô ình DenseNet121**
2. **Cài đặt mô hình**

****

Hình 24: Cài đặt mô hình DenseNet121

**BatchNormalization:** Lớp chuẩn hóa để cải thiện tốc độ huấn luyện và độ ổn định.

**DenseNet121(...):** Tải mô hình DenseNet121 đã được huấn luyện trước trên ImageNet.

**Input\_shape :**Kích thước đầu vào ảnh .

**Include\_top=False:** Bỏ các lớp fully connected gốc (head cũ), để bạn tùy biến phần phân loại cuối.

**Weights='imagenet':** Sử dụng trọng số đã được huấn luyện sẵn trên tập ImageNet.

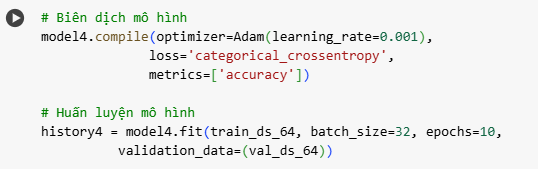
**Flatten():** Biến đầu ra từ CNN thành vector 1D để đưa vào fully connected.

**Dense(1024, activation='relu'):** Lớp ẩn mạnh với 1024 nơ-ron, dùng ReLU để học phi tuyến.

**Dropout(0.5)**: Giảm overfitting bằng cách tắt ngẫu nhiên 50% nơ-ron trong quá trình học.

**Dense(len(classes), activation='softmax'):** Lớp phân loại đầu ra (số lượng lớp = len(classes)), dùng softmax để trả về xác suất mỗi lớp.

1. **Huấn luyện mô hình**



Hình 25: Huấn luyện mô hình

**model4.compile(...):** Lệnh biên dịch mô hình trước khi huấn luyện, bao gồm việc chỉ định hàm tối ưu, hàm mất mát và các chỉ số đánh giá.

**learning\_rate=0.001:** Tốc độ học cố định. Giá trị này vừa đủ nhanh để mô hình học hiệu quả, vừa tránh việc “quá đà” dẫn tới không hội tụ.

**oss='categorical\_crossentropy':** Hàm mất mát dùng cho bài toán phân loại nhiều lớp (multi-class classification) với nhãn được mã hóa one-hot.

**metrics=['accuracy']:** Sử dụng độ chính xác (accuracy) làm chỉ số đánh giá trong suốt quá trình huấn luyện và kiểm tra.

**model4.fit(...):** Bắt đầu quá trình huấn luyện mô hình.

**train\_ds\_64:** Tập dữ liệu huấn luyện, thường được định dạng dưới dạng tf.data.Dataset hoặc Numpy array.

**batch\_size=32:** Chia dữ liệu thành các lô nhỏ (batch) gồm 32 mẫu trước khi đưa vào mô hình.

**epochs=10:** Số lần quét toàn bộ dữ liệu huấn luyện.

**history4:** Kết quả trả về từ fit() là một đối tượng History, chứa thông tin chi tiết về quá trình huấn luyện như: loss, accuracy, val\_loss, val\_accuracy qua từng epoch.

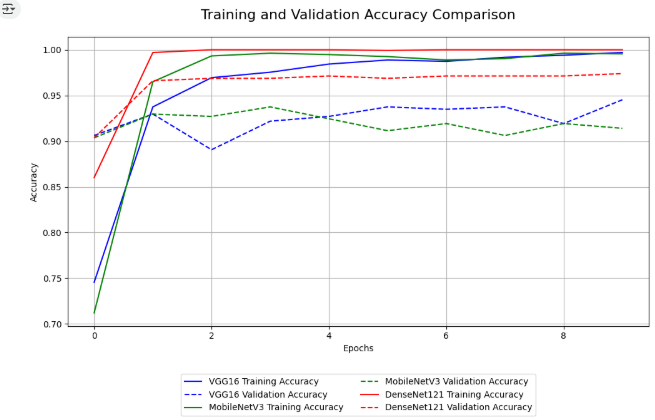
Trong 10 vòng huấn luyện của mô hình DenseNet121, thời gian huấn luyện trung bình của mỗi vòng là khoảng 436 giây, tương đương hơn 07 phút. Độ lỗi thấp nhất đạt được trên tập đánh giá là 0.0897 và độ chính xác cao nhất trên tập đánh giá là 0.9740, có nghĩa là mô hình dự đoán đúng 97.40% số hình ảnh của tập kiểm thử. Mô hình cho thấy khả năng học rất tốt với độ chính xác gần như tuyệt đối trên tập huấn luyện và độ chính xác cao, ổn định trên tập kiểm thử.

1. **Đánh giá hiệu suất và so sánh**

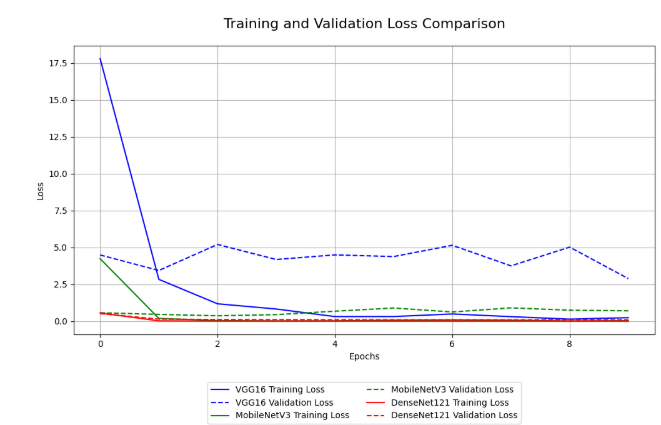
Để chọn ra được mô hình mang lại hiệu suất hiệu quả và toàn diện nhất, nhóm đã tiến hành so sánh 3 mô hình với nhau. Các mô hình này đều được huấn luyện trên cùng một tập dữ liệu và được đánh giá trên cùng một tập kiểm thử.

1. **So sánh Accuracy và Loss**

Chúng tôi so sánh độ chính xác (accuracy) và độ mất mát (loss) của các mô hình trên tập kiểm thử. Độ chính xác là tỷ lệ phần trăm các dự đoán đúng trên tổng số dự đoán, trong khi độ mất mát đo lường sự khác biệt giữa các giá trị dự đoán của mô hình và các giá trị thực tế.



Hình 26: Biểu đồ so sánh độ chính xác trong huấn luyện và kiểm tra



Hình 27: Biểu đồ so sánh độ mất mát trong huấn luyện và kiểm tra.

|  |  |  |  |
| --- | --- | --- | --- |
| Mô hình | VGG16 | MobileNetV3 | DenseNet121 |
| Độ chính xác huấn luyện | Tăng nhanh chóng và đạt đỉnh gần 1.0 ở epoch 2 | Tăng đều nhưng thấp hơn so với VGG16 , đạt khoảng 0.9 ở epoch 2 và duy trì | Tăng rất nhanh, đạt đỉnh gần 1.0 ở epoch 1 và duy trì mức độ cao nhất |
| Độ chính xác kiểm tra | Tăng dần nhưng dao động quanh mức 0.9 từ epoch 2 đến epoch 9. | Tăng ban đầu sau đó dao động, nhưng có xu hướng giảm nhẹ từ epoch 2 đến 9 | Tăng đều và ổn định, dao động nhẹ quanh mức 0.95 từ epoch 2 đến 9 |
| Độ mất mát huấn luyện | Giảm nhanh chóng từ epoch 0 đến epoch 1, sau đó duy trì ổn định | Giảm nhanh chóng và duy trì ổn định ở mức thấp sau epoch 1 | Giảm nhanh chóng và duy trì ổn định ở mức thấp nhất. |
| Độ mất mát kiểm tra | Giảm ban đầu nhưng sau đó tăng nhẹ và dao động, không ổn định. | Duy trì ổn định sau khi giảm ban đầu, không dao động nhiều. | Duy trì ở mức thấp và ổn định, ít dao động |

Bảng 1: Kết luận so sánh hiệu suất các mô hình học sâu trong huấn luyện và kiểm tra.

Nhận xét tổng quát:

* **DenseNet121** cho thấy kết quả tốt nhất với độ chính xác cao và ổn định cả trong quá trình huấn luyện và kiểm tra, cùng với độ mất mát thấp và ít dao động.
* **VGG16** cũng có hiệu suất khá tốt nhưng có một số dao động trong độ chính xác kiểm tra và độ mất mát kiểm tra.
* **MobileNetV3** cho thấy dao động lớn hơn trong độ chính xác kiểm tra và độ mất mát kiểm tra, có thể ít ổn định hơn trong các tập dữ liệu kiểm tra

1. **So sánh Accuracy, Precision, Recall, F1-score**

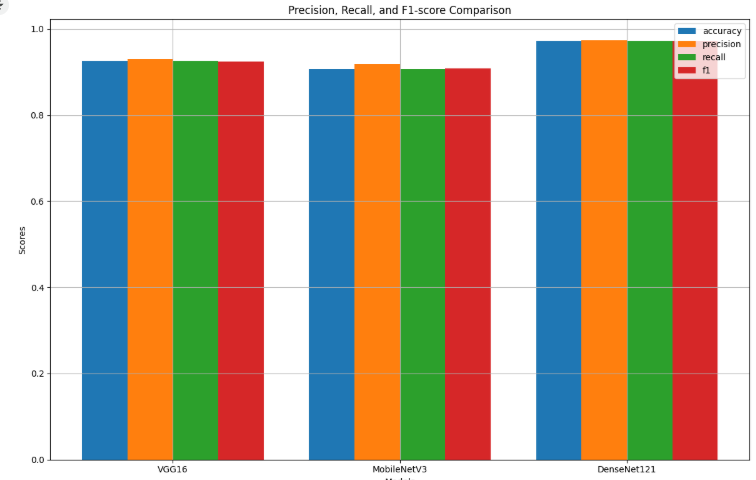
Các chỉ số khác như precision, recall và F1-score cũng được sử dụng để đánh giá

hiệu suất của mô hình:

* **Precision**: Tỷ lệ các dự đoán đúng trong số các dự đoán là đúng.
* **Recall**: Tỷ lệ các dự đoán đúng trong số các mẫu thực sự đúng.
* **F1-score**: Trung bình hài hòa của precision và recall.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score |
| VGG16 | 0,9259 | 0,9304 | 0,9259 | 0,9247 |
| MobileNetV3 | 0,9074 | 0,9178 | 0,9074 | 0,9079 |
| DenseNet121 | 0,0972 | 0,9737 | 0,9722 | 0,9723 |

Bảng 2: So sánh các chỉ số hiệu suất của các mô hình học sâu.



Hình 28: Biểu đồ so sánh các chỉ số độ chính xác, độ chính xác dự đoán, độ nhạy và F1-score của các mô hình.

Dựa vào Bảng trên nhóm có một số nhận xét:

**DenseNet121:**

* Đây là mô hình có hiệu suất tốt nhất trong số các mô hình được so sánh, với độ chính xác (Accuracy), độ chính xác (Precision), độ nhạy (Recall), và điểm F1 (F1-score) đều đạt 0.9723.
* Điều này cho thấy DenseNet121 không chỉ dự đoán chính xác mà còn duy trì cân bằng tốt giữa độ chính xác và độ nhạy.

**VGG16:**

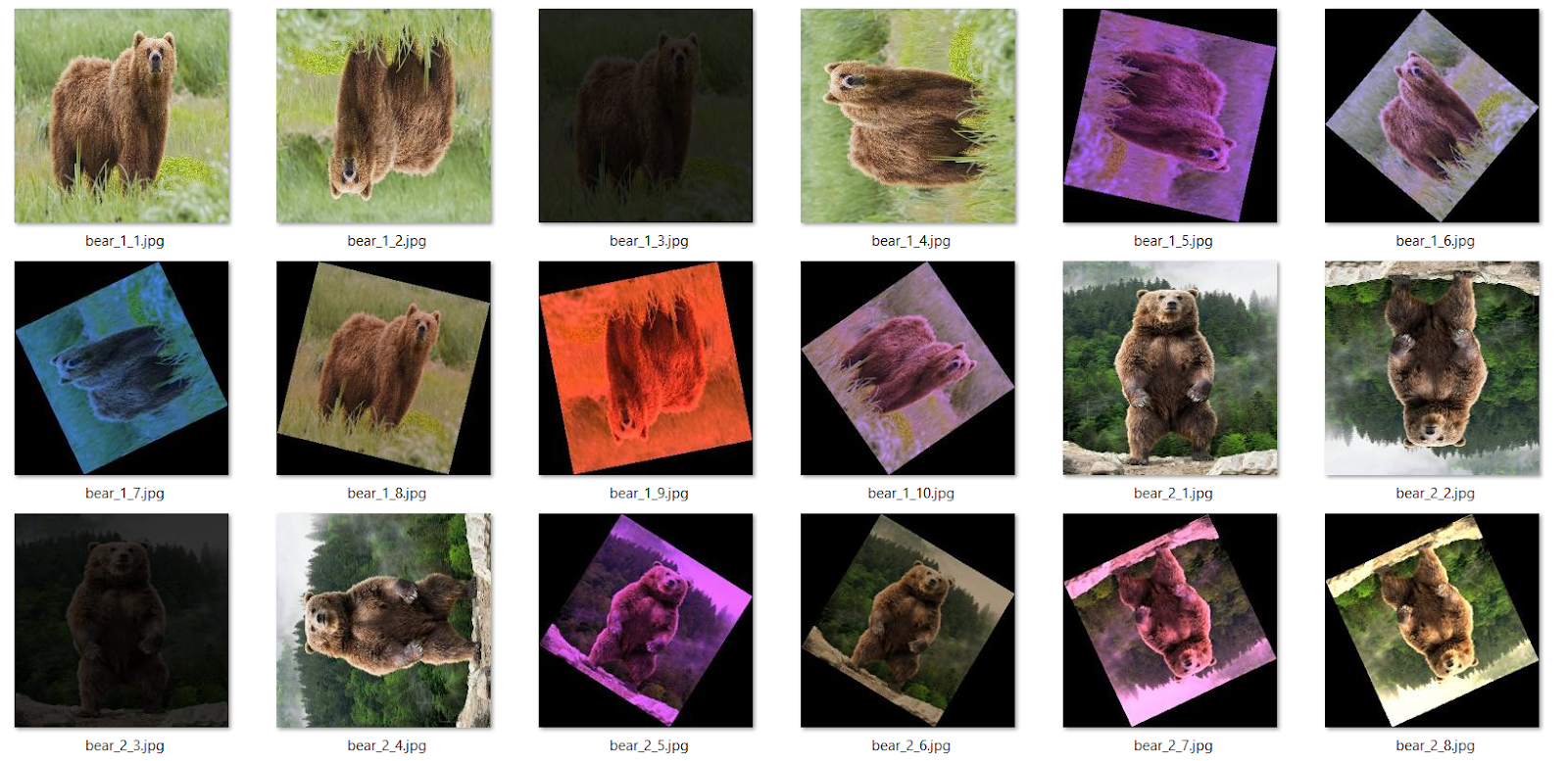
* Mô hình này cũng có hiệu suất rất cao với các chỉ số Accuracy, Precision, Recall và F1-score đều xấp xỉ 0.9247.
* So với DenseNet121, VGG16 có hiệu suất thấp hơn một chút, nhưng vẫn rất mạnh mẽ và đáng tin cậy.

**MobileNetV3:**

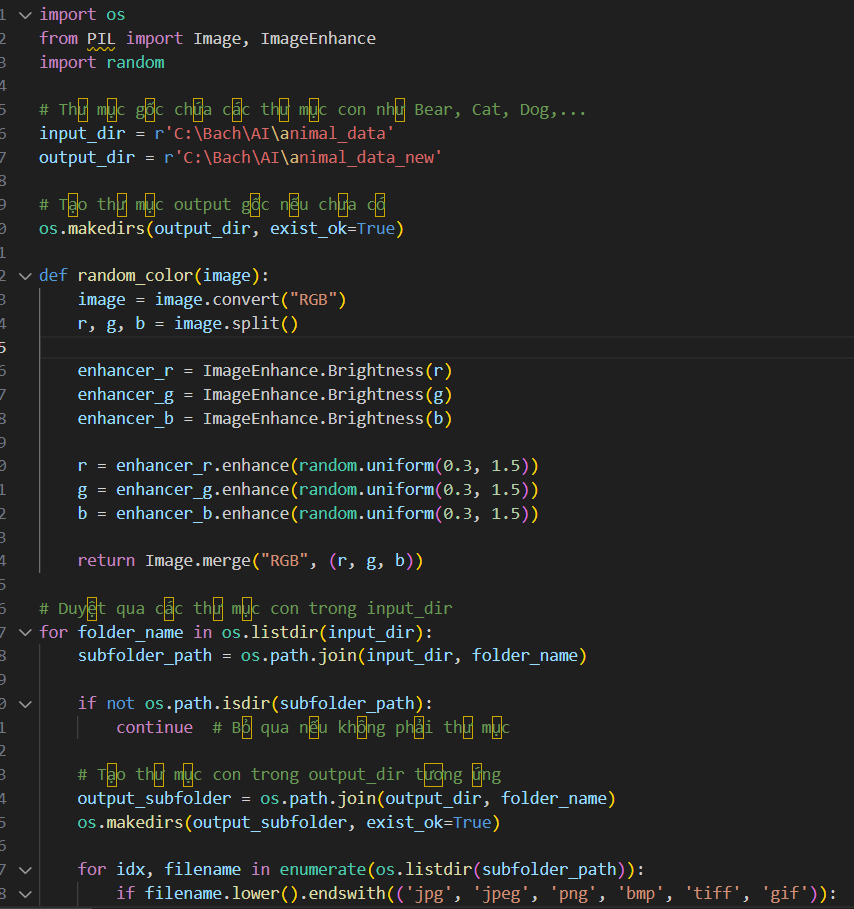
* Đây là mô hình có hiệu suất thấp nhất trong số các mô hình được so sánh, với Accuracy và Recall chỉ đạt 0.9074 và F1-score là 0.9079.
* Tuy nhiên, MobileNetV3 có ưu điểm là nhẹ và nhanh, phù hợp cho các ứng dụng yêu cầu tài nguyên tính toán thấp hoặc cần triển khai trên thiết bị di động
* Tóm lại thì model DenseNet121 cho hiệu suất cao nhất trong việc training

1. **Đề xuất cải tiến**
2. **Tăng cường dữ liệu (Data Augmentation)**

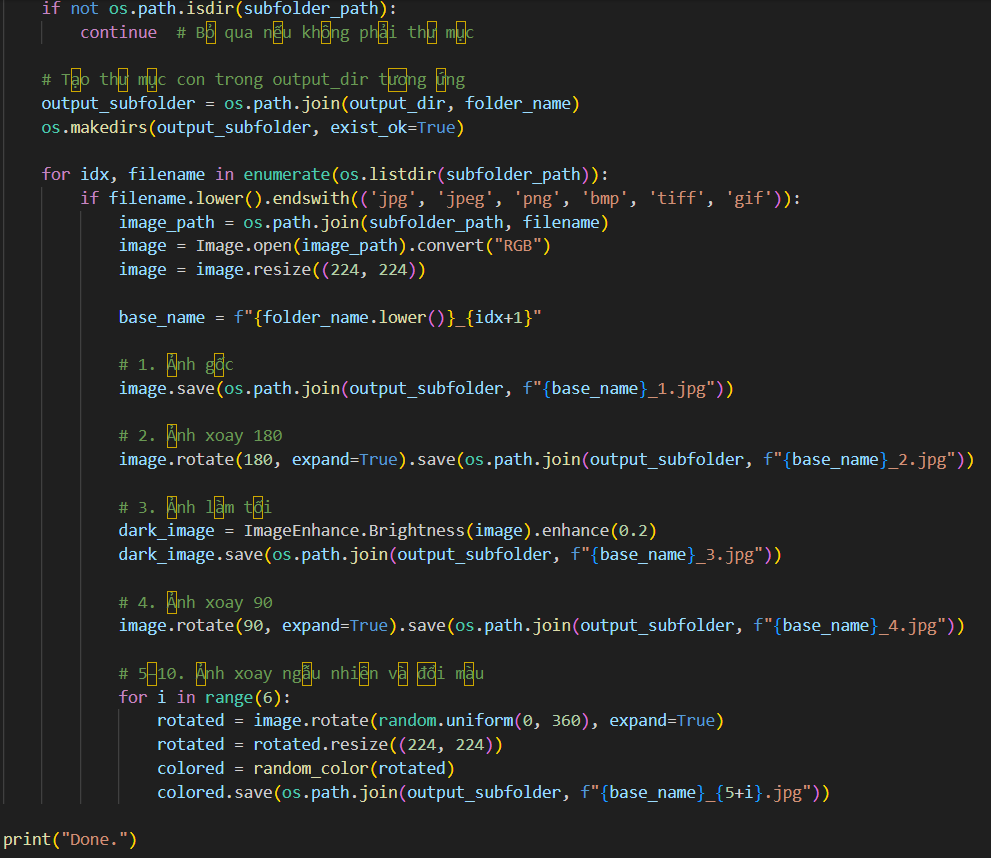
Để mô hình học sâu hoạt động hiệu quả và không bị overfitting, nhóm đã áp dụng các kỹ thuật tăng cường dữ liệu. Các kỹ thuật này bao gồm tô đen, tăng màu, xoay, và thay đổi màu sắc của ảnh để tạo ra các biến thể khác nhau từ ảnh gốc. Những thay đổi này giúp tạo ra nhiều mẫu huấn luyện khác nhau từ một ảnh gốc, làm phong phú thêm tập dữ liệu và cải thiện khả năng tổng quát hóa của mô hình.



Hình 29: Tăng cường dữ liệu



Hình 30: Code tăng cường dữ liệu 1



Hình 31: Code tăng cường dữ liệu 2

Đoạn code này thực hiện tăng cường dữ liệu ảnh cho các thư mục con chứa ảnh động vật. Cụ thể:

* Đọc ảnh từ thư mục input\_dir, resize về kích thước (224, 224) và lưu vào output\_dir.
* Với mỗi ảnh, tạo ra nhiều biến thể như: giữ nguyên, xoay 180°, làm tối, xoay 90°, và xoay ngẫu nhiên + đổi độ sáng màu.
* Hàm random\_color() giúp thay đổi độ sáng của từng kênh màu RGB một cách ngẫu nhiên.
* Kết quả được lưu thành nhiều ảnh mới trong thư mục tương ứng với mỗi lớp.
* Đây là kỹ thuật phổ biến để mở rộng tập dữ liệu huấn luyện mô hình học sâu.

1. **Thêm Early Stopping và Model Checkpoint**



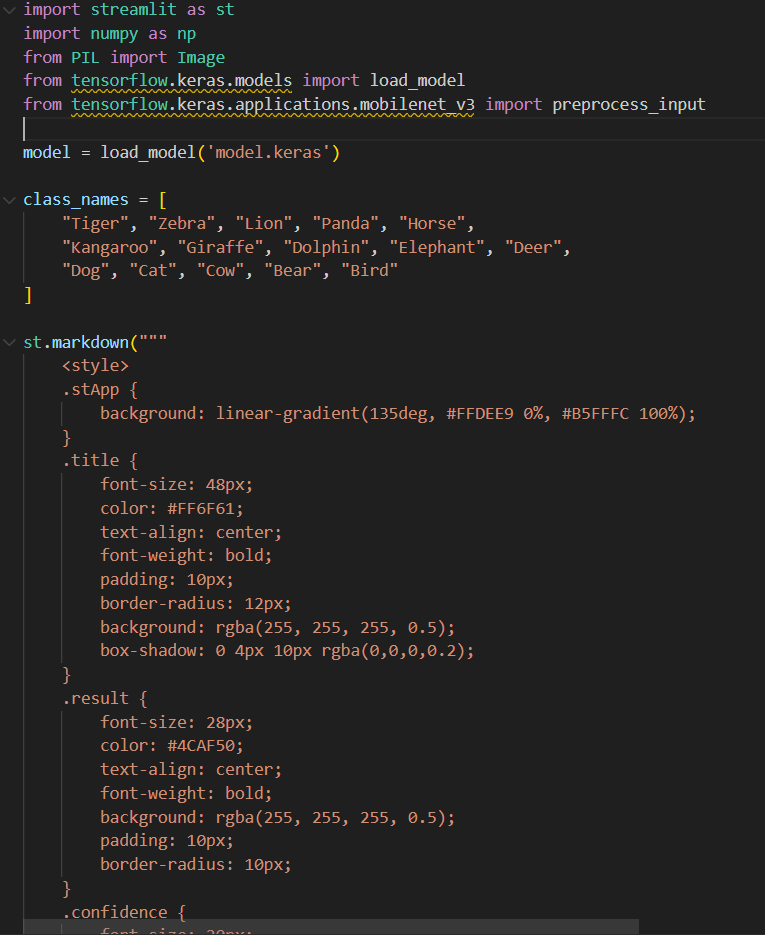
Hình 32: Code kĩ thuật  Early Stopping và Model Checkpoint

Trong quá trình huấn luyện mô hình học sâu, việc lựa chọn số epoch tối ưu là một thách thức quan trọng. Nếu huấn luyện quá ít epoch, mô hình sẽ chưa học đủ các đặc trưng cần thiết (underfitting), trong khi nếu huấn luyện quá nhiều epoch, mô hình có nguy cơ học quá mức dữ liệu huấn luyện (overfitting), làm giảm khả năng tổng quát hoá với dữ liệu mới. Để giải quyết vấn đề này, nhóm đã tích hợp hai kỹ thuật: EarlyStopping và ModelCheckpoint.

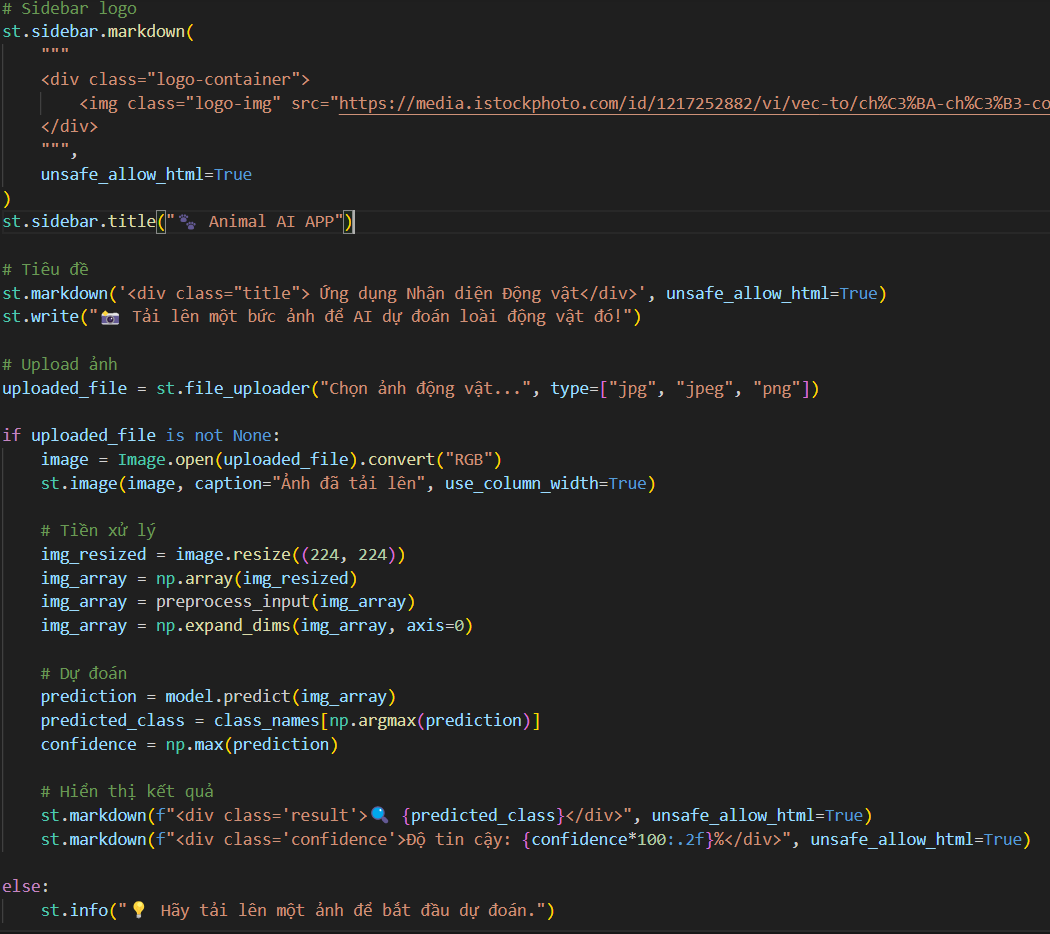
* EarlyStopping giúp tự động dừng quá trình huấn luyện nếu sau một số epoch nhất định mà độ chính xác trên tập kiểm tra (hoặc độ mất mát) không còn được cải thiện nữa. Điều này giúp tiết kiệm thời gian huấn luyện, tài nguyên tính toán, đồng thời tránh hiện tượng overfitting.
* ModelCheckpoint cho phép lưu lại trọng số của mô hình tốt nhất trong quá trình huấn luyện, dựa trên một tiêu chí xác định . Nhờ đó, sau khi huấn luyện kết thúc, mô hình sẽ sử dụng trọng số tốt nhất đạt được thay vì trọng số ở epoch cuối cùng, giúp cải thiện hiệu suất dự đoán.

Việc kết hợp cả hai kỹ thuật này không chỉ giúp tối ưu hóa quá trình huấn luyện mà còn tăng độ chính xác của mô hình khi áp dụng vào các dữ liệu thực tế.

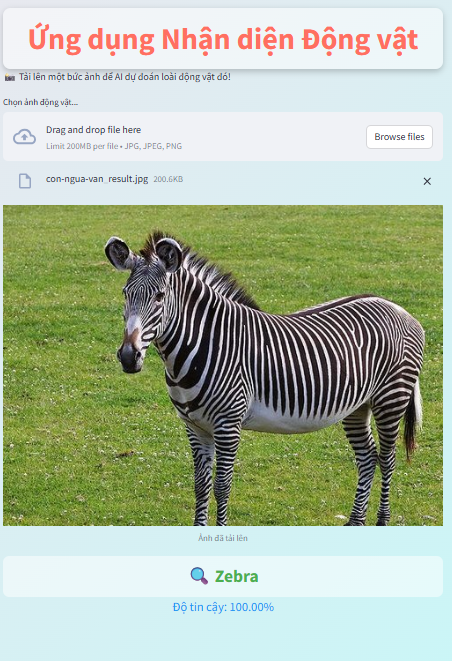
1. **Dự đoán hình ảnh bằng Streamlet với mô hình MobileNet**



Hình 33: Code ứng dụng nhận diện động vật 1



Hình 34: Code ứng dụng nhận diện động vật 2



Hình 35: Kết quả dự đoán hình ảnh bằng Streamlet với mô hình MobileNet

**KẾT LUẬN**

Trong quá trình phát triển và huấn luyện mô hình nhận diện động vật sử dụng Computer Vision, nhóm chúng em đã đạt được nhiều kết quả tích cực và đồng thời cũng nhận ra những điểm hạn chế cần cải thiện trong tương lai.

Về mặt kỹ thuật, nhóm đã triển khai thành công các mô hình học sâu như **VGG16**, **MobileNetV3Small** và **DenseNet121** để giải quyết bài toán phân loại ảnh động vật. Trong đó, **DenseNet121** thể hiện hiệu suất tốt nhất với độ chính xác lên tới **97.40%**, cùng các chỉ số Precision, Recall và F1-score đều đạt mức cao và ổn định. Các đồ thị về độ chính xác và độ mất mát trong quá trình huấn luyện cũng cho thấy mô hình đã học tốt và không xảy ra hiện tượng overfitting. Mô hình **MobileNetV3Small**, dù có hiệu suất thấp hơn một chút, nhưng nổi bật với khả năng tính toán nhanh, phù hợp với các ứng dụng triển khai trên thiết bị di động. Ngoài ra, nhóm cũng đã xây dựng một ứng dụng nhận diện động vật bằng **Streamlit**, cho phép người dùng tải ảnh và nhận kết quả dự đoán một cách trực quan.

Bên cạnh đó, nhóm đã áp dụng các kỹ thuật quan trọng như **tăng cường dữ liệu** (data augmentation), **early stopping** và **model checkpoint** để cải thiện chất lượng mô hình, giảm thời gian huấn luyện và tăng độ chính xác trên dữ liệu thực tế.

Tuy nhiên, nhóm cũng gặp phải một số **hạn chế** đáng chú ý. Thứ nhất, dữ liệu huấn luyện vẫn còn hạn chế về tính đa dạng; nhiều ảnh có màu sắc, bố cục hoặc góc chụp tương đồng, dẫn đến mô hình khó phân biệt giữa các loài động vật có ngoại hình gần giống nhau. Thứ hai, mô hình vẫn còn nhầm lẫn trong một số trường hợp, đặc biệt khi ảnh đầu vào bị mờ, thiếu sáng hoặc có nhiều đối tượng chồng lấn. Cuối cùng, thời gian huấn luyện của mô hình **DenseNet121** tương đối dài, điều này có thể gây khó khăn khi triển khai trên hệ thống có giới hạn tài nguyên.

Trong tương lai, nhóm sẽ tiếp tục cải tiến mô hình bằng cách:

* Thu thập và bổ sung thêm dữ liệu huấn luyện đa dạng hơn về điều kiện ánh sáng, góc chụp, môi trường
* Thử nghiệm với các mô hình mới như EfficientNet hoặc các kiến trúc transformer cho thị giác máy tính
* Áp dụng **ensemble learning** để kết hợp các mô hình và nâng cao độ chính xác
* Tối ưu hóa siêu tham số và mở rộng ứng dụng sang video nhận diện động vật theo thời gian thực.

**TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | Salesforce, 2025. [Trực tuyến]. Available: https://www.tableau.com/data-insights/ai/algorithms. |
| [2] | M. K. S. Likhon, 2024. [Trực tuyến]. Available: https://www.kaggle.com/datasets/likhon148/animal-data/data. |
| [3] | AWS, 2023. [Trực tuyến]. Available: https://aws.amazon.com/vi/what-is/computer-vision/. |
| [4] | V. An, 2018. [Trực tuyến]. Available: https://quantrimang.com/cong-nghe/thi-giac-may-tinh-computer-vision-la-gi-154485. |
| [5] | NguyenThiHueC, 2020. [Trực tuyến]. Available: https://viblo.asia/p/thi-giac-may-tinh-la-gi-cac-van-de-lien-quan-toi-thi-giac-may-tinh-Az45b4M6ZxY. |
| [6] | A. Sarkar, 2020. [Trực tuyến]. Available: https://medium.com/data-science/creating-densenet-121-with-tensorflow-edbc08a956d8. |
| [7] | A. G. H. v. c. sự, 2017. [Trực tuyến]. Available: https://arxiv.org/abs/1704.04861. |
| [8] | D. V. S. v. c. sự, 2022. [Trực tuyến]. Available: https://thegioitailieuso.dlib.vn/tailieuvn/doc/nhan-dien-bien-bao-giao-thong-viet-nam-thoi-gian-thuc-bang-thuat-toan-mobilenet-ket-hop-ssd-tren-boa-2532138.html. |
| [9] | K. &. Z. A. Simonyan, 2014. [Trực tuyến]. Available: https://arxiv.org/abs/1409.1556. |
| [10] | AYOUB, 2022. [Trực tuyến]. Available: https://www.linkedin.com/pulse/everything-you-need-know-vgg16-ayoub-kirouane. |
| [11] | TensorFlow, 2022. [Trực tuyến]. Available: https://www.tensorflow.org/tutorials/images/transfer\_learning?hl=vi. |