BỘ GIÁO DỤC & ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA CHẤT LƯỢNG CAO

----------⸸Δ⸸----------

BÁO CÁO CUỐI KÌ

NHẬN DIỆN 20 LOÀI THÚ CƯNG SỬ DỤNG REAL-TIME

MÔN : Trí Tuệ Nhân Tạo

GVHD: Nguyễn Trường Thịnh

SVTH : Nguyễn Thanh Pháp

MSSV : 19146230

Ngành : CNKT Cơ Điện Tử

Lớp : Thứ 7, Tiết 12-15

Tp. Hồ Chí Minh tháng 06 năm 2022

LỜI CẢM ƠN

Sau một thời gian học tập và nghiên cứu đề tài "Nhận diện và tracking 20 loài thú cưng" em đã không ngừng được củng cố, mở mang thêm nhiều kiến thức và tầm hiểu biết về lĩnh vực Trí Tuệ Nhân Tạo qua sự dạy dỗ, đào tạo tận tình của các thầy cô bộ môn trong khoa Đào Tạo Chất Lượng Cao và sự học hỏi lẫn nhau giữa bạn bè cùng khóa. Thông qua bản báo cáo này, em xin được gửi lời cảm ơn chân thành đến Thầy Nguyễn Trường Thịnh đã tạo những điều kiện tốt nhất cho chúng em trong suốt thời gian học tập vừa qua, hướng dẫn chúng em tận tình trong suốt quá trình thực hiện đề tài này. Mặc dù cố gắng hoàn thành đồ án trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của Thầy và tất cả các bạn để đề tài ngày càng hoàn thiện hơn. Em xin trân trọng cảm ơn!

Sinh viên,

Nguyễn Thanh Pháp

MSSV: 19146230

MỤC LỤC

PHỤ LỤC HÌNH ẢNH

Chương 1: Tổng Quan Đề Tài

1.1 Tính cấp thiết của đề tài

Trong thời đại 4.0, con người áp dụng công nghệ trí tuệ nhân tạo trong mọi lĩnh vực của cuộc sống để tiết kiệm thời gian và công sức.

Trong lĩnh vực giáo dục, các bài giảng của thầy cô giáo cần lặp đi lặp lại nhiều lần tên những loài động vật nói chung cũng như các loài thú cưng thường nuôi nói riêng rất nhiều lần để giúp học sinh nhớ lâu và không quên, đặc biệt là trẻ từ 3-6 tuổi. Với ứng dụng nhận diện thú cưng này, sẽ khắc phục được tình trạng đó, giảm bớt gánh nặng, hoạt động cho giáo viên, ngoài việc là công cụ giảng dạy tạo niềm cảm hứng học tập mới cho trẻ nhỏ, ứng dụng còn có thể xem video thế giới động vật hay tranh ảnh truyền thống phù hợp với quá trình học chữ của trẻ nhỏ từ những loài động vật, thú cưng trong nhà, không những thế ứng dụng này còn giúp các bậc phụ huynh kèm cặp con trẻ tại nhà.

Việc nhận diện thế giới quan từ 3 đến 6 tuổi là vô cùng quan trọng chính vì vậy em nhìn thấy việc tạo ra ứng dụng này là vô cùng quan trọng và cấp thiết.

Hình 1 Nhận diện thú cưng đối với trẻ nhỏ

Trong thời gian gần đây, nhờ có sự phát triển mạnh mẽ về khả năng tính toán của các thế hệ máy tính hiện đại cũng như sự bùng nổ dữ liệu thông qua mạng lưới Internet trải rộng, ta đã chứng kiến nhiều sự đột phá trong lĩnh vực Trí tuệ nhân tạo, Mechine learning cũng như là deep learning. Sự quay lại và phát triển vượt bậc của phương pháp học sâu bằng thuật toán CNN đã có được những thành tựu to lớn trong lĩnh vực nhận diện ảnh, trong đó có bài toán nhận diện động vật. Đề tài nghiên cứu “Nhận diện 20 loài thú cưng sử dụng realtime” đã được đưa ra với hy vọng có thể ứng dụng thành công các mô

hình học sâu hiện đại để xây dựng các hệ thống nhận diện động vật một cách tự động, đặc biệt là đối với các loại thú cưng thường nuôi ở nước ta.

1.2 Mục tiêu của đề tài

1.3 Cấu trúc của đề tài

Dựa trên mục tiêu cụ thể đã trình bày trong phần trước, đề tài được tổ chức thành năm chương với các nội dung cụ thể như sau:

- **Chương 1:** Trong chương tổng quan này, ta sẽ có ra cái nhìn tổng quan về các hướng tiếp cận và giải pháp đã được ứng dụng trong bài toán nhận diện các loài vật nuôi, từ các phương pháp thuần tính toán xử lý ảnh tương đối thô sơ cho tới các phương pháp Học máy truyền thống và cuối cùng là các phương pháp Học sâu.
- **Chương 2:** Trong chương tiếp theo, ta sẽ mô tả tổng quan Hệ thống nhận diện vật nuôi tự động do em phát triển, với các mô đun chính như máy chủ, máy trạm, bộ huấn luyện và nhận diện, bộ tiền xử lý... Ngoài ra, cách thức thu thập, chỉnh sửa cơ sở dữ liệu ảnh và cách cài đặt, triển khai môi trường huấn luyện cho mô hình mạng nơ-ron tích chập và tìm hiểu chìa khóa giải quyết bài toán nhận diện ảnh với bộ dữ liệu huấn luyện có kích thước tương đối nhỏ.
- **Chương 3:** Chương 3 tập trung trình bày về kết quả thực nghiệm, bao gồm kết quả so sánh độ chính xác giữa các phương pháp Học máy truyền thống với phương pháp Học sâu, cùng với các đánh giá về độ hiệu quả của bộ tạo dữ liệu ảnh nhiều cũng như các ảnh chụp ứng dụng khi được sử dụng trong thực tế. Dựa trên các kết quả thực nghiệm này, ta sẽ đưa ra một số phân tích và kết luận về điểm mạnh và điểm hạn chế của mô hình huấn luyện Học sâu đã chọn.Cuối cùng, sẽ tổng kết các nội dung đã trình bày trong đề tài, từ đó đề xuất các phương hướng nghiên cứu tiếp theo để tiếp tục cải thiện chất lượng nhận diện của hệ thống.

1.4 Bài toán nhận diện thú cưng bằng thuật toán CNN

Nhận diện vật thể trong ảnh được coi là bài toán cơ bản nhất trong lĩnh vực Trí tuệ nhân tạo, là nền tảng cho rất nhiều bài toán mở rộng khác như bài toán phân lớp, định vị, tách biệt vật thể.... Tuy bài toán cơ bản này đã tồn tại hàng thế kỷ nhưng con người vẫn chưa thể giải quyết nó một cách triệt để, do tồn tại rất nhiều khó khăn để máy tính có thể hiểu được các thông tin trong một bức ảnh. Những khó khăn tiêu biểu có thể kể đến: sự đa dạng điểm nhìn, sự đa dạng kích thước, điều kiện ánh sách khác biệt, sự ẩn giấu vật thể sau các đối tượng khác trong ảnh, sự lẫn lộn với nền và sự đa dạng về chủng loại vật

thể… Là một trường hợp cụ thể của bài toán nhận diện và phân lớp, bài toán nhận diện các loài vật nuôi kế thừa các khó khăn vốn có của bài toán gốc, và kèm theo là các khó khăn riêng của chính nó, như: số lượng khổng lồ về chủng loài vật nuôi theo vùng miền, địa hình, khí hậu… với vô số loài vật nuôi có hình dáng, kết cấu giống nhau hay sự đa dạng về hình dạng của cùng một loài vật do ảnh hưởng của thời tiết, cách chăm sóc của từng người chủ cũng như chế độ dinh dưỡng…

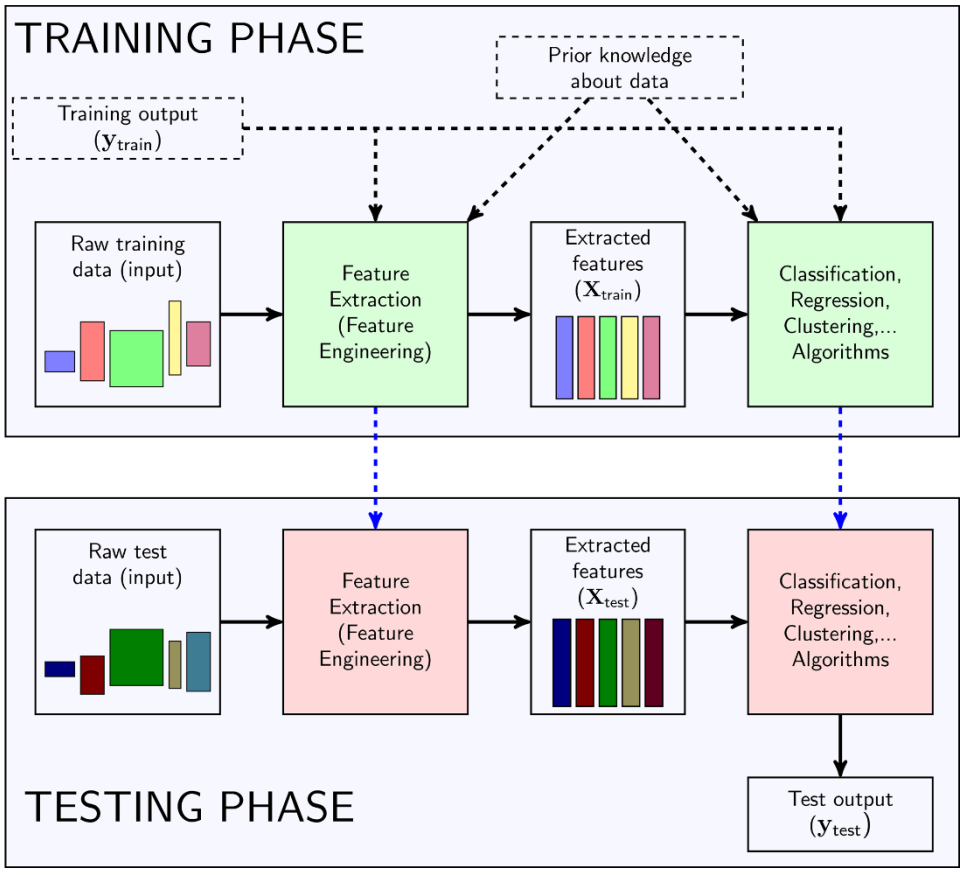
1.4.1 Các hướng tiếp cận và giải quyết bài toán

Bài toán tự động nhận diện vật nuôi (thú cưng) đã xuất hiện từ lâu và đã có rất nhiều bài báo, công trình khoa học được đưa ra nhằm đề xuất hoặc cải tiến các thuật toán nhận diện. Trong đó, xuất hiện sớm nhất là các phương pháp Xử lý ảnh – Image Processing, các phương pháp này tập trung vào phát triển các thuật toán nhằm trích xuất thông tin, ví dụ các tham số về màu sắc, hình dạng, kết cấu, kích thước…, từ bức ảnh đầu vào để nhận diện con vật. Do chỉ đơn thuần xử lý trên một vài ảnh đầu vào trong khi sự biến thiên về màu sắc, hình dạng, kích thước… của vật nuôi quá phức tạp, kết quả đạt được của các phương pháp này không được cao và phạm vi áp dụng trên số lượng loài động vật cũng bị hạn chế.

Bắt đầu từ những năm 2000s, sau khi xuất hiện một vài bài bài báo khoa học đề xuất áp dụng phương pháp Học máy - Machine Learning - vào bài toán nhận diện vật nuôi với độ chính xác cao, hướng giải quyết bài toán đã tập trung vào ứng dụng và cải tiến các thuật toán Học máy, cụ thể là nghiên cứu, thử nghiệm trích chọn các đặc trưng phù hợp nhất để đưa vào huấn luyện bộ nhận diện tự động. Trong những năm gần đây, nhờ sự phát triển vượt bậc về sức mạnh tính toán của các máy tính cũng như sự bùng nổ dữ liệu trên Internet, Học sâu - Deep Learning đã đạt được nhiều thành tựu đáng kể, đặc biệt là trong lĩnh vực Xử lý ảnh và ngôn ngữ tự nhiên. Học sâu cũng đã được áp dụng rất thành công vào bài toán nhận diện các loài động vật, trong phạm vi hạn chế về số lượng loại vật nuôi cần nhận diện, phương pháp này đã đạt được kết quả cao đáng kinh ngạc.

1.4.2 Phương pháp Machine Learning truyền thống

Mô hình hoạt động chung của các phương pháp Machine Learning được thể hiện trong hình dưới đây:



Hình 2 Mô hình hoạt động của phương pháp Mechine learning

Ta có thể thấy Machine Learning gồm hai giai đoạn chính là Huấn luyện – Training và Thử nghiệm – Testing, trong mỗi giai đoạn đều sử dụng hai thành phần quan trọng nhất do người xử lý bài toán thiết kế, đó là Trích chọn đặc trưng – Feature Engineering (hay còn gọi là Feature Extraction) và Thuật toán phân loại, nhận diện… - Algorithms. Hai thành phần này có ảnh hưởng trực tiếp đến kết quả bài toán, vì thế được thiết kế rất cẩn thận, tốn nhiều thời gian, đòi hỏi người thiết kế phải có kiến thức chuyên môn và nắm rõ đặc điểm của bài toán cần xử lý.

Trích chọn đặc trưng - Feature Engineering

Trong các bài toán thực tế, ta chỉ có được những dữ liệu thô chưa qua chọn lọc xử lý, và để có thể đưa các dữ liệu này vào huấn luyện ta cần có những phép biến đổi để biến các dữ liệu thô thành dữ liệu chuẩn, không còn nhiễu và có khả năng biểu diễn dữ liệu tốt hơn. Các thông tin đặc trưng này là khác nhau với từng loại dữ liệu và bài toán cụ thể, vì thế trong từng trường hợp phép biến đổi này cần phải được tùy biến một cách thích hợp để cải thiện độ chính xác của mô hình dự đoán. Quá trình này được gọi là Trích chọn đặc trưng

1.4.3 Phương pháp Học sâu (Deep learning)

Là 1 lĩnh vực chuyên sâu của Học máy, đã xuất hiện từ những năm 1980s nhưng chưa phổ biến cho đến thập kỷ gần đây do các nhà khoa học đã có thể tận dụng khả năng tính toán mạnh mẽ của các máy tính hiện đại cũng như khối lượng dữ liệu khổng lồ (hình ảnh, âm thanh, văn bản,…) trên Internet. Các mạng huấn luyện theo phương pháp Học sâu còn được gọi với cái tên khác là mạng nơ-ron sâu (Deep Neural Network) do cách thức hoạt động của chúng. Về cơ bản, các mạng này bao gồm rất nhiều lớp khác nhau, mỗi lớp sẽ phân tích dữ liệu đầu vào theo các khía cạnh khác nhau và theo mức độ trừu tượng nâng cao dần.

Chương 2. Xây Dựng Hệ Thống

2.1 Xây dựng bộ dữ liệu

Do ở ngoài tự nhiên có hàng trăm loài vật nuôi nên em sẽ chọn ra 20 loài vật nuôi được xem là thú cưng mà các gia đình Việt Nam ưa chuộng vào mô hình. Danh sách các loài vật nuôi mà em lựa chọn và phân loại trong tập dữ liệu ảnh bao gồm như sau: Cá Betta, Chuột Hamster, Gà tre cảnh, Lợn Cảnh, Nhện Tarantula, Nhím Kiểng, Rùa, Tắc Kè, Thỏ, Vẹt, Chó, Mèo, Cá Hải Tượng, Đại Bàng, Công, Rắn Ngô,……Tổng số ảnh cho mỗi loài động vật là 200 ảnh, tổng số ảnh trong cả bộ dữ liệu là 4000 ảnh, em chia tập dữ liệu ra làm 3 phần Train, Validation và Test, mỗi phần đều có hình ảnh của 20 loài, trong đó:

+ Tổng số ảnh trong tập Train là 2800 ảnh, mỗi loài động vật là 140 ảnh, tỉ lệ train chiếm 70%.

+ Tổng số ảnh trong tệp Validation là 800 ảnh mỗi loài động vật là 40 ảnh, tỉ lệ Validation chiếm 20%.

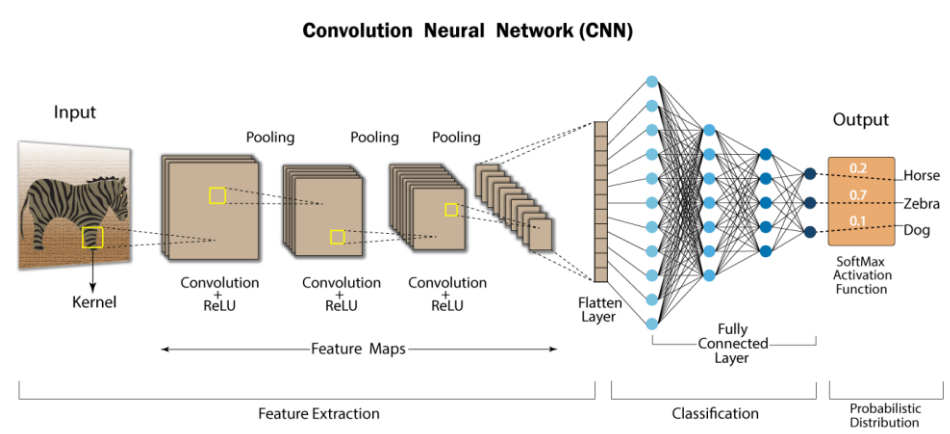
+ Tổng số ảnh trong tập Test là 400 ảnh, mooic loài động vật là 20 ảnh, tỉ lệ Test chiếm 10%.

2.2 Sử dụng Mạng nơ-ron tích chập CNN

Em sẽ sử dụng mô hình mạng CNN huấn luyện tập dữ liệu để nhận diện các loài vật nuôi. Mạng nơ-ron tích chập (CNN - Convolutional Neural Network) là một trong những mô hình mạng Học sâu phổ biến nhất hiện nay, có khả năng nhận diện và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon… cho các mục đích khác nhau như các thuật toán tagging tự động, tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

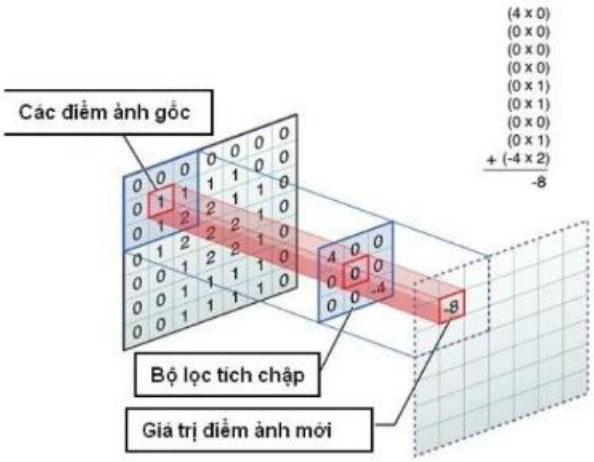
2.2.1 Kiến trúc mạng nơ-ron tích chập

Các lớp cơ bản trong một mạng CNN bao gồm: Lớp Convolutional, Lớp ReLU (Rectified Linear Unit), Lớp Pooling và Lớp Fully-connected, được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau.



Hình 3 Kiến trúc mạng nơron tích chập

- Lớp Convolutional (tích chập): Đây là thành phần quan trọng nhất trong mạng CNN, cũng là nơi thể hiện tư tưởng xây dựng sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ này được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc – filters – có kích thước nhỏ.



Hình 4 Phép tích chập các giá trị điểm ảnh

Trong ví dụ, ta thấy bộ lọc được sử dụng là một ma trận có kích thước 3x3. Bộ lọc này được dịch chuyển lần lượt qua từng vùng ảnh đến khi hoàn thành quét toàn bộ bức ảnh, tạo ra một bức ảnh mới có kích thước nhỏ hơn hoặc bằng với kích thước ảnh đầu vào. Như vậy, sau khi đưa một bức ảnh đầu vào cho lớp Tích chập ta nhận được kết quả đầu ra là một loạt ảnh tương ứng với các bộ lọc đã được sử dụng để thực hiện phép tích chập. Các trọng số của các bộ lọc này được khởi tạo ngẫu nhiên trong lần đầu tiên và sẽ được cải thiện dần xuyên suốt quá trình huấn luyện.

- Lớp ReLU: Lớp này được xây dựng với ý nghĩa đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loạt các phép tính toán tuyến tính qua các lớp

Tích chập. Trong số các hàm kích hoạt phổ biến nhất như tank, sigmoid…, hàm ReLU được chọn do cài đặt đơn giản, tốc độ xử lý nhanh mà vẫn đảm bảo được tính toán hiệu quả. Cụ thể, phép tính toán của hàm ReLU chỉ đơn giản là chuyển tất cả các giá trị âm thành giá trị 0.

Thông thường, lớp ReLU được áp dụng ngay phía sau lớp Tích chập, với đầu ra là một ảnh mới có kích thước giống với ảnh đầu vào, các giá trị điểm ảnh cũng hoàn toàn tương tự trừ các giá trị âm đã bị loại bỏ.

- Lớp Pooling: Một thành phần tính toán chính khác trong mạng CNN là pooling, thường được đặt sau lớp Tích chập và lớp ReLU để làm giảm kích thước kích thước ảnh đầu ra trong khi vẫn giữ được các thông tin quan trọng của ảnh đầu vào. Có hai phương pháp lấy mẫu phổ biến hiện nay là lấy mẫu giá trị lớn nhất (Max Pooling) và lấy mẫu giá trị trung bình (Avarage Pooling).

- Lớp Fully-connected: Lớp Fully-connected này hoàn toàn tương tự như trong mạng nơ-ron truyền thống, tức là tất cả các điểm ảnh được kết nối đầy đủ với node trong lớp tiếp theo.

2.2.2 Học chuyển giao và tinh chỉnh mô hình huấn luyện

Trong thời gian đầu khi các phương pháp Học sâu mới đạt được nhiều thành tựu và được áp dụng phổ biến, trong cộng đồng Học sâu đã xuất hiện một quan niệm sai lầm: nếu bạn không có lượng dữ liệu huấn luyện khổng lồ, bạn không thể tạo ra một mô hình dự đoán hiệu quả.

Học chuyển giao – Transfer Learning – chính là từ khóa để giải quyết điểm hạn chế này. Học chuyển giao là quá trình khai thác, tái sử dụng các tri thức đã được học tập bởi một mô hình huấn luyện trước đó vào giải quyết một bài toán mới mà không phải xây dựng một mô hình huấn luyện khác từ đầu.

Hiện nay, phương pháp phổ biến thường được áp dụng với trường hợp bộ CSDL tương đối nhỏ là tận dụng một mạng CNN đã được huấn luyện trước đó với bộ dữ liệu rất lớn như ImageNet - Mạng CNN này sẽ chỉ được sử dụng như một bộ trích chọn đặc trưng cho bộ CSDL huấn luyện mới, bằng cách thay thế các lớp Fully-connected ở cuối mạng và giữ cố định các tham số cho toàn bộ các lớp còn lại của mạng. - Không chỉ thay thế và huấn luyện lại bộ nhận diện cuối cùng của mạng CNN, mà đồng thời ta thực hiện tối ưu, tinh chỉnh (Fine-tune) một vài hoặc tất cả các lớp trong mạng. Ý tưởng của việc tái sử dụng mạng CNN là dựa trên nhận định rằng các đặc trưng được học trong các lớp đầu của

mạng là các đặc trưng chung nhất, hữu dụng với phần lớn bài toán, ví dụ: đặc trưng về cạnh, hình khối hay các khối màu… Các lớp sau đó của mạng CNN sẽ nâng dần độ cụ thể, riêng biệt của các chi tiết phục vụ cho bài toán nhận diện ban đầu. Do đó, ta hoàn toàn có thể tái sử dụng lại các lớp đầu của mạng CNN mà không phải mất nhiều thời gian và công sức huấn luyện từ đầu.

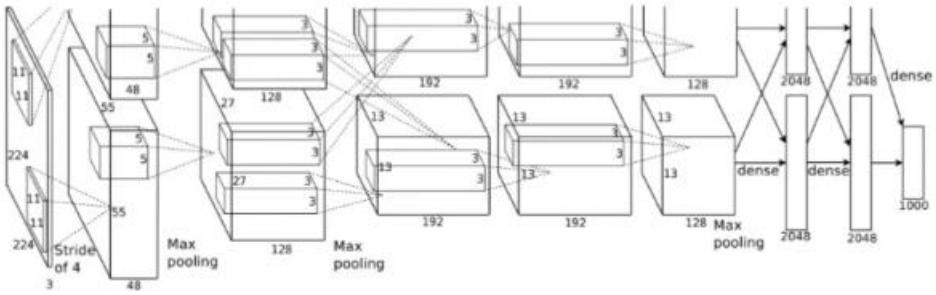
2.2.3. Mạng huấn luyện AlexNet

Mạng huấn luyện AlexNet là công trình đầu tiên phổ biến mạng CNN trong lĩnh vực Thị giác máy tính, cũng là một trong những mạng huấn luyện CNN nổi tiếng nhất nhờ thành tích ấn tượng mà nó đạt được trong cuộc thi nhận diện ảnh quy mô lớn (ILSVC) tổ chức vào năm 2012.

Nhiệm vụ chính của cuộc thi đặt ra mà các đội tham gia phải giải quyết là bài toán nhận diện, với bộ dữ liệu huấn luyện lên đến 1,2 triệu ảnh được gán nhãn cho 1.000 hạng mục khác nhau. Nhóm SuperVision, gồm các thành viên Alex Krizhevsky, Ilya Sutskever và Geoff Hinton, cùng với mạng AlexNet của họ đã đạt được kết quả đáng kinh ngạc là chiến thắng áp đảo nhóm đứng thứ hai với độ chính xác chênh lệch đến hơn 10% (15,31% và 26,17%). Điều kinh ngạc là mạng huấn luyện này chỉ nhận dữ liệu đầu vào là các giá trị điểm ảnh thô và không hề áp dụng bất kỳ phương pháp trích chọn đặc trưng nào, trong khi mọi hệ thống nhận giác thị giác truyền thống đều phải gồm nhiều giai đoạn trích chọn đặc trưng hết sức tỉ mỉ, cẩn thận, thậm chí phải áp dụng nhiều mẹo để cải thiện chất lượng nhận diện.

Kiến trúc mạng AlexNet

Nhóm của Alex Krizhevsky đã công bố một bài báo với tiêu đề “ImageNet Classification with Deep Convolutional Networks”, đưa ra mô tả cụ thể về kiến trúc của mạng AlexNet cũng như cách thức cài đặt và sử dụng các lớp trong mạng để huấn luyện mô hình với bộ dữ liệu ảnh khổng lồ của ImageNet. Mạng có cấu trúc tương đối đơn giản nếu so với các mạng CNN hiện đại gần đây, bao gồm 5 lớp Convolutional và 3 lớp Fully-connected, được huấn luyện song song trên 2 card đồ họa GPU.



Hình 5 Kiến trúc mạng AlexNet

Ứng dụng mạng AlexNet vào bài toán Nhận diện vật nuôi

a) Cài đặt mạng AlexNet với một mô hình đã được huấn luyện trước với bộ ảnh của ImageNet.

b) Xây dựng bộ CSDL ảnh huấn luyện

c) Tinh chỉnh lại mô hình để giải quyết bài toán nhận diện 20 loài động vật. Mặc dù kích thước CSDL ảnh không lớn, song độ chính xác của mô hình nhận diện vẫn được đảm bảo nhờ khả năng trích chọn đặc trưng tự động của mạng AlexNet.

2.3 Các bước chạy mô hình huấn luyện trên Google Colab

Bước 1: Khai báo các thư viện cần sử dụng, bao gồm:

Tensorflow: là một thư viện có mã nguồn mở được dùng để tính toán mechinery với quy mô lớn có khả năng tương thích và quy mô lớn.

Numpy: là một thư viện toán học phổ biến của python hooxx trợ cho việc tính toán các mảng và ma trận đa chiều có kích thước lớn với các hàm đã được tối ưu, áp dụng lên các mảng đa chiều đó với mục đích xử lí dữ liệu một cách nhanh chóng.

Matplotlib: là một trong số các thư viện Python phổ biến nhất được sử dụng trực quang hóa dữ liệu.

Sklearn.model_selection import train_test_split: Thư viện này để chia dữ liệu thành tập huấn luyện và tập thử nghiệm.

Os: cho phép thao tác với tệp hoặc thư mục.

Keras.models: sử dụng khi mô hình thích hợp cho một lớp đơn giản.

Tensorflow.keras.optimizers: đào tạo và đánh giá bằng các phương pháp tích hợp, API chức năng, độ chính xác.

Keras.callbacks: xây dựng một mô hình mạng noron sâu cho một bài toán phân loại.

Sklearn.utils: kiểm tra xem công cụ ước tính có phù hợp hay không bằng cách xác minh sự hiện diện của các thuộc tính phù hợp và nếu không nó sẽ tăng NotFittedError.

Preprocessing: cung cấp một số chức năng tiện ích phổ biến và các lớp biến áp để thay đổi các vector đặc trưng thô thành một vector phù hợp cho việc xây dựng model.

 # Code khai báo thư viện:

```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import os
import numpy as np
import tensorflow as tf
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.optimizers import SGD, RMSprop
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.utils import validation
from sklearn import preprocessing
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Bước 2: Tiền xử lí dữ liệu

```
#Chuyển đến thư mục có chứa hình ảnh thú cưng
%cd /content/drive/MyDrive/Colab Notebooks/cuoi_ki/THUCUNG
# Tạo ra class ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)
```

```
# Tạo bộ dữ liệu training set
training_set=train_datagen.flow_from_directory('/content/drive/MyDrive/Colab
Notebooks/cuoi_ki/THUCUNG/Train',target_size=(150,150), batch_size=32,
class_mode='categorical')
```

```
Found 2879 images belonging to 20 classes.
```

```
    # Tạo bộ dữ liệu validation
validation=train_datagen.flow_from_directory('/content/drive/MyDrive/Colab Notebooks/cuoi_ki/THUCUNG/Validation',target_size=(150,150), batch_size=32, class_mode='categorical')
```

```
Found 806 images belonging to 20 classes.
```

Bước 3: Địa chỉ của các lớp.

```
# Các nhãn có trong bộ dữ liệu training set
training_set.class_indices
# Các nhãn có trong bộ dữ liệu validation
validation.class_indices
```

```
{'Ca Betta': 0,
 'Ca Hai Tuong': 1,
 'Chao Mao': 2,
 'Cho Alaska': 3,
 'Cho Poodle': 4,
 'Chuot Hamster': 5,
 'Cong': 6,
 'Dai Bang': 7,
 'Ga Tre Canh': 8,
 'Heo': 9,
 'Khi Duoi Soc': 10,
 'Meo Munchkin': 11,
 'Nhen Tarantula': 12,
 'Nhim Kieng': 13,
 'Ran Ngo': 14,
 'Rua': 15,
 'Soc Kieng': 16,
 'Tac Ke': 17,
 'Tho': 18,
 'Vet': 19}
```

Bước 4: Thiết lập model

```
    # Tạo ra mạng CNN để train mô hình
model=Sequential()
    # Tích chập 32 lần với mỗi lần là 3 hàng 3 cột
model.add(Conv2D(32,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same',input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
    # Lấy phần tử lớn nhất ở trong 2 hàng và 2 cột
```

```
model.add(MaxPooling2D(2,2))
    # Tích chập 64 lần với mỗi lần là 3 hàng 3 cột
model.add(Conv2D(64,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
    # Lấy phần tử lớn nhất ở trong 2 hàng và 2 cột
model.add(MaxPooling2D(2,2))
    # Tích chập 128 lần với mỗi lần là 3 hàng 3 cột
model.add(Conv2D(128,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
    # Lấy phần tử lớn nhất ở trong 2 hàng và 2 cột
model.add(MaxPooling2D(2,2))
    # Tích chập 256 lần với mỗi lần là 3 hàng 3 cột
model.add(Conv2D(256,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(256,(3,3), activation='relu', kernel_initializer='he_uniform', padding='same'))
    # Lấy phần tử lớn nhất ở trong 2 hàng và 2 cột
model.add(MaxPooling2D(2,2))
    # Duỗi thẳng dữ liệu
model.add(Flatten())
    # Tạo lớp ẩn thứ nhất với 128 tín hiệu ra
model.add(Dense(512,activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
    # Tạo lớp ẩn thứ hai với 20 tín hiệu ra
model.add(Dense(20,activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 150, 150, 32)      896

 conv2d_9 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d_4 (MaxPooling  (None, 75, 75, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 75, 75, 64)        18496

 conv2d_11 (Conv2D)          (None, 75, 75, 64)        36928

 max_pooling2d_5 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_12 (Conv2D)          (None, 37, 37, 128)       73856

 conv2d_13 (Conv2D)          (None, 37, 37, 128)       147584

 max_pooling2d_6 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 conv2d_14 (Conv2D)          (None, 18, 18, 256)       295168

 conv2d_15 (Conv2D)          (None, 18, 18, 256)       590080

 max_pooling2d_7 (MaxPooling  (None, 9, 9, 256)        0
 2D)

 flatten_1 (Flatten)         (None, 20736)             0

 dense_3 (Dense)             (None, 512)               10617344

 dropout_2 (Dropout)         (None, 512)               0

 dense_4 (Dense)             (None, 256)               131328

 dropout_3 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 20)                5140

=================================================================
Total params: 11,926,068
Trainable params: 11,926,068
Non-trainable params: 0
_________________________________________________________________
```

Bước 5: Biên dịch

```
# Biên dịch
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
# Gán biến lại để vẽ đồ thị, với 100 lần học (epochs), mỗi lần học thì chỉ học 64 dữ liệu (batch_size), khi sai số không thay đổi trong 20 lần học thì sẽ dừng học
history=model.fit(training_set,epochs=100,batch_size=64,verbose=1, validation_data=validation, callbacks=[EarlyStopping(monitor='val_loss', patience=20)])
```

```
Epoch 1/100
90/90 [==============================] - 785s 9s/step - loss: 3.0003 - accuracy: 0.0788 - val_loss: 2.9738 - val_accuracy: 0.0844
Epoch 2/100
90/90 [==============================] - 58s 642ms/step - loss: 2.8788 - accuracy: 0.1153 - val_loss: 2.8039 - val_accuracy: 0.1141
Epoch 3/100
90/90 [==============================] - 59s 652ms/step - loss: 2.7644 - accuracy: 0.1469 - val_loss: 2.6624 - val_accuracy: 0.1725
Epoch 4/100
90/90 [==============================] - 57s 638ms/step - loss: 2.6341 - accuracy: 0.1966 - val_loss: 2.5989 - val_accuracy: 0.2097
Epoch 5/100
90/90 [==============================] - 58s 647ms/step - loss: 2.5292 - accuracy: 0.2188 - val_loss: 2.4463 - val_accuracy: 0.2556
Epoch 6/100
90/90 [==============================] - 58s 646ms/step - loss: 2.4274 - accuracy: 0.2546 - val_loss: 2.3096 - val_accuracy: 0.3102
Epoch 7/100
90/90 [==============================] - 59s 659ms/step - loss: 2.2893 - accuracy: 0.2966 - val_loss: 2.1395 - val_accuracy: 0.3598
Epoch 8/100
90/90 [==============================] - 58s 644ms/step - loss: 2.1588 - accuracy: 0.3359 - val_loss: 1.9143 - val_accuracy: 0.4293
Epoch 9/100
90/90 [==============================] - 59s 653ms/step - loss: 1.9931 - accuracy: 0.3838 - val_loss: 1.8111 - val_accuracy: 0.4442
Epoch 10/100
90/90 [==============================] - 60s 667ms/step - loss: 1.8823 - accuracy: 0.4161 - val_loss: 1.6628 - val_accuracy: 0.4950
Epoch 11/100
90/90 [==============================] - 58s 639ms/step - loss: 1.7486 - accuracy: 0.4575 - val_loss: 1.4407 - val_accuracy: 0.5422
Epoch 12/100
90/90 [==============================] - 59s 653ms/step - loss: 1.6057 - accuracy: 0.5123 - val_loss: 1.2946 - val_accuracy: 0.5955
Epoch 13/100
90/90 [==============================] - 58s 639ms/step - loss: 1.4713 - accuracy: 0.5446 - val_loss: 1.1866 - val_accuracy: 0.6526
Epoch 14/100
90/90 [==============================] - 60s 662ms/step - loss: 1.3739 - accuracy: 0.5773 - val_loss: 1.0012 - val_accuracy: 0.7084
Epoch 15/100
```

```
90/90 [==============================] - 58s 648ms/step - loss: 0.1195 - accuracy: 0.9687 - val_loss: 0.0360 - val_accuracy: 0.9876
Epoch 87/100
90/90 [==============================] - 61s 675ms/step - loss: 0.1079 - accuracy: 0.9673 - val_loss: 0.0206 - val_accuracy: 0.9963
Epoch 88/100
90/90 [==============================] - 63s 700ms/step - loss: 0.1315 - accuracy: 0.9604 - val_loss: 0.0221 - val_accuracy: 0.9938
Epoch 89/100
90/90 [==============================] - 59s 657ms/step - loss: 0.0970 - accuracy: 0.9729 - val_loss: 0.0178 - val_accuracy: 0.9938
Epoch 90/100
90/90 [==============================] - 59s 657ms/step - loss: 0.1012 - accuracy: 0.9733 - val_loss: 0.0501 - val_accuracy: 0.9839
Epoch 91/100
90/90 [==============================] - 60s 672ms/step - loss: 0.0797 - accuracy: 0.9736 - val_loss: 0.0406 - val_accuracy: 0.9901
Epoch 92/100
90/90 [==============================] - 59s 655ms/step - loss: 0.1106 - accuracy: 0.9684 - val_loss: 0.0451 - val_accuracy: 0.9864
Epoch 93/100
90/90 [==============================] - 60s 665ms/step - loss: 0.1574 - accuracy: 0.9583 - val_loss: 0.0522 - val_accuracy: 0.9851
Epoch 94/100
90/90 [==============================] - 59s 661ms/step - loss: 0.1110 - accuracy: 0.9680 - val_loss: 0.0728 - val_accuracy: 0.9801
Epoch 95/100
90/90 [==============================] - 59s 656ms/step - loss: 0.1156 - accuracy: 0.9691 - val_loss: 0.0288 - val_accuracy: 0.9913
Epoch 96/100
90/90 [==============================] - 60s 669ms/step - loss: 0.1409 - accuracy: 0.9597 - val_loss: 0.0181 - val_accuracy: 0.9950
Epoch 97/100
90/90 [==============================] - 58s 643ms/step - loss: 0.0716 - accuracy: 0.9795 - val_loss: 0.0081 - val_accuracy: 0.9975
Epoch 98/100
90/90 [==============================] - 58s 645ms/step - loss: 0.1135 - accuracy: 0.9625 - val_loss: 0.0496 - val_accuracy: 0.9826
Epoch 99/100
90/90 [==============================] - 57s 634ms/step - loss: 0.1520 - accuracy: 0.9576 - val_loss: 0.0354 - val_accuracy: 0.9913
Epoch 100/100
90/90 [==============================] - 57s 632ms/step - loss: 0.1073 - accuracy: 0.9673 - val_loss: 0.0182 - val_accuracy: 0.9938
```

Bước 6: Kiểm tra kết quả, đánh giá độ chính xác, vẽ đồ thị

Đánh giá độ chính xác của mô hình

```
Score=model.evaluate(training_set,verbose=0)
print('Train Loss', Score[0])
print('Train Accuracy', Score[1])
```

Vẽ đồ thị giữa số lần học (Epochs) và độ chính xác (Accuracy)

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train','Validation'])
plt.show
```

Lưu lại

```
model.save("thucung.h5")
```

Tải mô hình

```
model_CNN=load_model('thucung.h5')
```

```
test="/content/drive/MyDrive/Colab Notebooks/cuoi_ki/THUCUNG/Validation/Ca
Betta"
for i in os.listdir(test):
  img=load_img(test+'/'+i,target_size=(150,150))
  plt.imshow(img)
  img=img_to_array(img)
  img=img.astype('float32')
  img=img/255
  img=np.expand_dims(img,axis=0)
  result=model_CNN.predict(img)
  if round(result[0][0])==1:
    prediction='Ca Betta'
  if round(result[0][1])==1:
    prediction='Ca Hai Tuong'
  if round(result[0][2])==1:
    prediction='Chao Mao'
  if round(result[0][3])==1:
```

```
  prediction='Cho Alaska'
if round(result[0][4])==1:
  prediction='Cho Poodle'
if round(result[0][5])==1:
  prediction='Chuot Hamster'
if round(result[0][6])==1:
  prediction='Cong'
if round(result[0][7])==1:
  prediction='Dai Bang'
if round(result[0][8])==1:
  prediction='Ga Tre Canh'
if round(result[0][9])==1:
  prediction='Heo'
if round(result[0][10])==1:
  prediction='Khi Duoi Soc'
if round(result[0][11])==1:
  prediction='Meo Munchkin'
if round(result[0][12])==1:
  prediction='Nhen Tarantula'
if round(result[0][13])==1:
  prediction='Nhim Kieng'
if round(result[0][14])==1:
  prediction='Ran Ngo'
if round(result[0][15])==1:
  prediction='Rua'
if round(result[0][16])==1:
  prediction='Soc Kieng'
if round(result[0][17])==1:
  prediction='Tac Ke'
if round(result[0][18])==1:
  prediction='Tho'
if round(result[0][19])==1:
```

```
  prediction='Vet'
 print(prediction)
 plt.show()
```

Bước 8: Chạy realtime nhận diện

```
# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
        js_reply: JavaScript object containing image from webcam
  Returns:
        img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
       bytes: Base64 image byte string
  """
  # convert array into PIL image
```

```
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = '{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes
```

```
from IPython.display import display, Javascript, Image
# JavaScript to properly create our live video stream using our webcam as input

def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;

    function removeDom() {
       stream.getVideoTracks()[0].stop();
       video.remove();
       div.remove();
       video = null;
       div = null;
       stream = null;
```

```
    imgElement = null;
    captureCanvas = null;
    labelElement = null;
  }

  function onAnimationFrame() {
    if (!shutdown) {
      window.requestAnimationFrame(onAnimationFrame);
    }
    if (pendingResolve) {
      var result = "";
      if (!shutdown) {
        captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
        result = captureCanvas.toDataURL('image/jpeg', 0.8)
      }
      var lp = pendingResolve;
      pendingResolve = null;
      lp(result);
    }
  }

  async function createDom() {
    if (div !== null) {
      return stream;
    }
    div = document.createElement('div');
    div.style.border = '2px solid black';
    div.style.padding = '3px';
    div.style.width = '100%';
    div.style.maxWidth = '600px';
    document.body.appendChild(div);
```

```
const modelOut = document.createElement('div');
modelOut.innerHTML = "<span>Status:</span>";
labelElement = document.createElement('span');
labelElement.innerText = 'No data';
labelElement.style.fontWeight = 'bold';
modelOut.appendChild(labelElement);
div.appendChild(modelOut);

video = document.createElement('video');
video.style.display = 'block';
video.width = div.clientWidth - 6;
video.setAttribute('playsinline', '');
video.onclick = () => { shutdown = true; };
stream = await navigator.mediaDevices.getUserMedia(
   {video: { facingMode: "environment"}});
div.appendChild(video);

imgElement = document.createElement('img');
imgElement.style.position = 'absolute';
imgElement.style.zIndex = 1;
imgElement.onclick = () => { shutdown = true; };
div.appendChild(imgElement);

const instruction = document.createElement('div');
instruction.innerHTML =
   '<span style="color: red; font-weight: bold;">' +
   'Bấm vào video để dừng</span>';
div.appendChild(instruction);
instruction.onclick = () => { shutdown = true; };

video.srcObject = stream;
await video.play();
```

```
  captureCanvas = document.createElement('canvas');
  captureCanvas.width = 640; //video.videoWidth;
  captureCanvas.height = 480; //video.videoHeight;
  window.requestAnimationFrame(onAnimationFrame);

  return stream;
}
async function stream_frame(label, imgData) {
  if (shutdown) {
    removeDom();
    shutdown = false;
    return '';
  }

  var preCreate = Date.now();
  stream = await createDom();

  var preShow = Date.now();
  if (label != "") {
    labelElement.innerHTML = label;
  }

  if (imgData != "") {
    var videoRect = video.getClientRects()[0];
    imgElement.style.top = videoRect.top + "px";
    imgElement.style.left = videoRect.left + "px";
    imgElement.style.width = videoRect.width + "px";
    imgElement.style.height = videoRect.height + "px";
    imgElement.src = imgData;
  }
```

```
    var preCapture = Date.now();
    var result = await new Promise(function(resolve, reject) {
      pendingResolve = resolve;
    });
    shutdown = false;

    return {'create': preShow - preCreate,
            'show': preCapture - preShow,
            'capture': Date.now() - preCapture,
            'img': result};
  }
  '')

 display(js)

def video_frame(label, bbox):
 data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
 return data
```

```
%cd /content
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import numpy as np
import PIL
import io
import cv2
from keras.models import  load_model

# start streaming video from webcam
video_stream()
# label for video
```

```
label_html = 'Đang lấy hình ảnh...'
# initialze bounding box to empty
bbox = ''
count = 0

model_file_path = "/content/drive/MyDrive/Colab Notebooks/THUCUNG/thucung.h5"
vggmodel = load_model(model_file_path)

classes = ['Ca Betta',
 'Ca Hai Tuong',
 'Chao Mao',
 'Cho Alaska',
 'Cho Poodle',
 'Chuot Hamster',
 'Cong',
 'Dai Bang',
 'Ga Tre Canh',
 'Heo',
 'Khi Suoi Soc',
 'Meo Muchkin',
 'Nhen Tarantula',
 'Nhim Kieng',
 'Ran Ngo',
 'Rua',
 'Soc Kieng',
 'Tac Ke',
 'Tho',
 'Vet']
while True:
   # Đọc ảnh trả về từ JS
   js_reply = video_frame(label_html, bbox)
   if not js_reply:
```

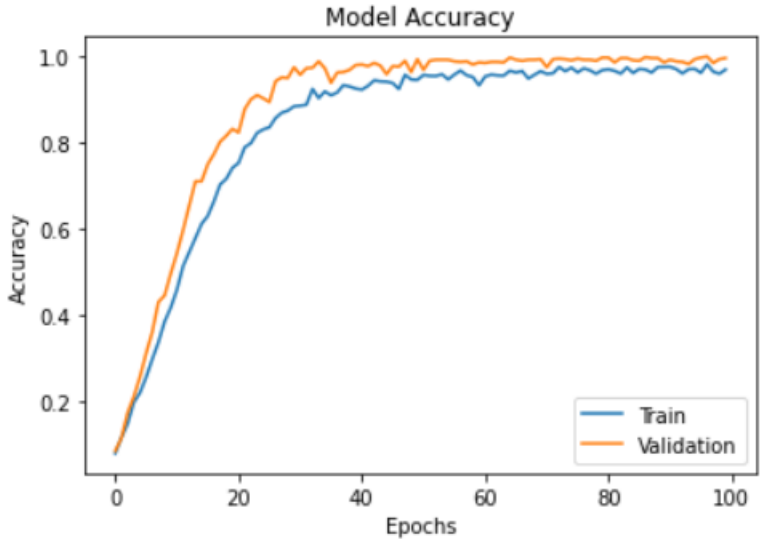
```
    break

# convert JS response to OpenCV Image
frame = js_to_image(js_reply["img"])

# Resize để đưa vào model
frame_p = cv2.resize(frame, dsize=(150,150))
tensor = np.expand_dims(frame_p, axis=0)

# Feed vào mạng
pred = vggmodel.predict(tensor)
class_id = np.argmax(pred)
class_name = classes[class_id]

# Vẽ lên một ảnh để tẹo nữa overlay

# create transparent overlay for bounding box
bbox_array = np.zeros([480,640,4], dtype=np.uint8)

bbox_array = cv2.putText(bbox_array, "{}".format(class_name),
                         (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                         (0, 255,0), 2)

bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
# convert overlay of bbox into bytes
bbox_bytes = bbox_to_bytes(bbox_array)
# update bbox so next frame gets new overlay
bbox = bbox_bytes
```

Chương 3. Kết quả thử nghiệm đánh giá, nhận xét và kết luận

3.1 Kết quả thực nghiệm

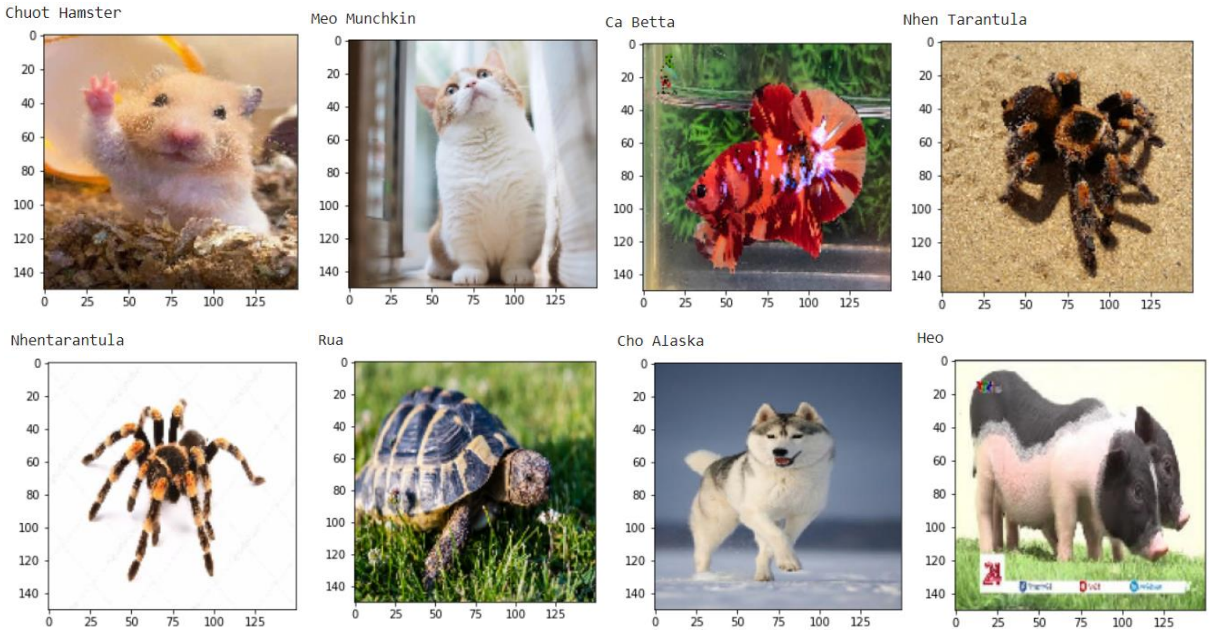
Với phương pháp Học sâu: Thống kê độ chính xác với tỉ lệ bộ Train/Validation/Test là 70/25/5. Độ chính xác đạt được là rất cao, ~99.16%, vượt trội so với với độ chính xác của mô hình huấn luyện sử dụng phương pháp Học máy truyền thống.

Biểu đồ thể hiện độ chính xác sau khi train mô hình:

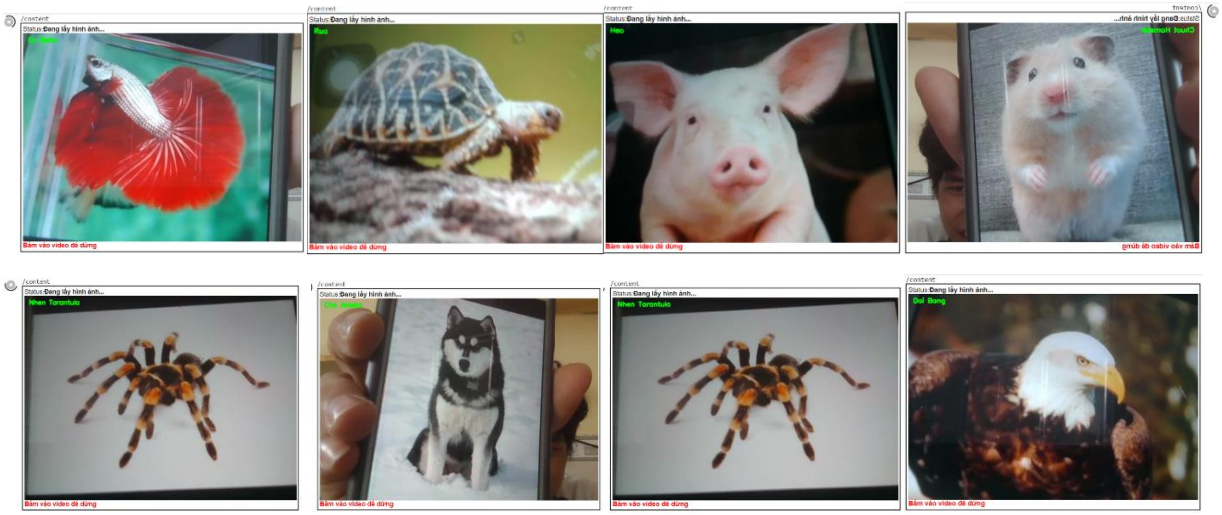


Hình 6 Biểu đồ thể hiện đồ chính xác của mô hình

Khi nhận diện các loài thú cưng bằng hình ảnh: Kết quả cho ra gần như chính xác tuyệt đối, có thể nhận diện chính xác hầu hết tất cả các ảnh.



Khi nhận diện các loài thú cưng bằng webcam sử dụng realtime: Kết quả cho ra có sự sai số kết quả với độ chính xác 82%.Khi để ảnh xa webcam nhận diện sai một vài hình nhưng khi để webcam lại gần thì nhận diện realtime hầu như là nhận diện được đúng tên loài thú cưng mà đã dạy trước.



3.2 Đánh giá kết quả

Với các bài toán nhận dạng và phân loại đối tượng nói chung, trong đó rất khó có thể chọn được các đặc trưng hiệu quả, thì Học sâu là phương pháp có ưu thế vượt trội so với các phương pháp Học máy truyền thống. Học sâu giúp đơn giản hóa quá trình huấn luyện mô hình nhận dạng khi không yêu cầu sự tham gia của người huấn luyện trong quá trình trích chọn đặc trưng, đồng thời cho phép tái sử dụng các mô hình đã huấn luyện trước để giảm thời gian cài đặt giải pháp cho các bài toán nhận dạng mới.

Sự cải thiện rõ rệt trong độ chính xác của mô hình nhận dạng sau khi tăng cường CSDL ảnh huấn luyện đã cho thấy hiệu quả thực tế của các phép sinh ảnh tự động sử dụng các phương pháp xử lý ảnh cơ bản. Chất lượng nhận dạng của ứng dụng trong thực tế cũng được tăng lên do các ảnh được sinh tự động giúp mô phỏng quá trình chụp ảnh trong đời thực, như các góc chụp khác nhau, các nhiễu sinh ra do môi trường, chất lượng máy ảnh… cũng như sự đa dạng của nền mà người dùng sử dụng để chụp ảnh. Việc tăng cường CSDL ảnh cũng là một giải pháp cho trường hợp khó thu thập ảnh để huấn luyện mô hình, tuy nhiên cũng cần phải chú ý đến mặt trái của việc lạm dụng phương pháp tăng cường ảnh này, đó là nguy cơ gây ra trạng thái “overfit” dữ liệu (mô hình nhận dạng quá khớp với dữ liệu huấn luyện mà bị sai lệch với dữ liệu thực tế).

3.3 Kết luận

- Đề tài project cuối kì môn Trí Tuệ Nhân Tạo đã nghiên cứu, tìm hiểu bài toán tự động nhận diện các loài vật nuôi, thú cưng trong ảnh, và thực hiện phát triển, cài đặt phương án giải quyết cho bài toán dựa trên sự tìm kiếm, thống kê các hướng tiếp cận đã được công bố qua rất nhiều bài báo, công trình khoa học trên thế giới. Các kết quả chính mà bài viết đã đạt được, tương ứng với các mục tiêu đề ra ban đầu như sau: Hoàn thiện xây dựng bộ cơ sở dữ liệu ảnh phục vụ huấn luyện nhận dạng cho 20 loài vật nuôi phổ biến ở nước ta, với số lượng ảnh gốc trung bình cho mỗi loại quả là từ 150-200 ảnh.
- Thống kê các đặc trưng thường được sử dụng để huấn luyện bộ nhận diện vật nuôi trong các phương pháp Học máy truyền thống, bao gồm các đặc trưng về màu sắc, hình dạng và kết cấu. Từ đó làm cơ sở xây dựng một mạng nơ-ron nhân tạo truyền thống và so sánh kết quả với một mạng nơ-ron tích chập thuộc nhóm phương pháp Học sâu.
- Cài đặt và tinh chỉnh một mạng nơ-ron tích chập đã được huấn luyện trước, ứng dụng vào bài toán nhận diện các loài vật nuôi.. Thực nghiệm với bộ dữ liệu test và trong thực tế đã cho kết quả khá tốt, nguyên nhân chính là do phạm vi số lượng vật nuôi để nhận dạng đã được hạn chế chỉ còn 20 loài
– một con số rất khiêm tốn so với số lượng thú cưng ở Việt Nam nói riêng và cả thế giới nói chung. Hệ thống tự động nhận dạng thú cưng còn cần rất nhiều cải thiện, đặc biệt là về khả năng mở rộng phạm vi loài vật cũng như kích thước, chất lượng của bộ CSDL ảnh huấn luyện.

3.4 Tổng kết

Môn học:

- Thông qua việc môn học trí tuệ nhân tạo, em đã được học rất nhiều điều mà nó có thể giúp ích cho công việc cũng như cuộc sống của em trong tương lai.

- Em đã được hiểu rõ hơn về trí tuệ nhân tạo cũng nhưng những ứng dụng và lợi ích mà nó sẽ mang lại cho cuộc sống con người.

Project cuối kì:

- Hoàn thành những công việc đã được giao
  + Bài báo cáo đề tài 30 trang
  + Bài báo về giải thuật xây dựng (giới thiệu, phương pháp, kết quả) : 6-10 trang
  + Poster giấy A4
  + Chạy real- time sử dụng webcam

+ Sử dụng machine learning, Deep learning, không sử dụng các thuật toán thuần xử lý ảnh, được sử dụng các thư viện cơ bản, không sử dụng các file pre-train.

- Hiểu rõ hơn về những việc mà ngành AI sẽ làm
- Em hi vọng với project cuối kì này sẽ góp phần phát triển ngành trí tuệ nhân tạo trong tương lại.

Một số thiếu sót: Chạy realtime chưa đạt đến mức độ chính xác trên 90%, chưa tạo được giao diện, mobile app, web,…để nhận diện các loài thú cưng. Trong quá trình chạy data mắc nhiều lỗi nên thời gian hoàn thành lâu phải sửa lại data nhiều lần. Xem cách làm giao diện web nhiều ngày nhưng vẫn chưa thực hiện được.

TÀI LIỆU THAM KHẢO

[1] https://www.slideshare.net/trongthuy3/luan-van-nhan-dang-va-phan-loai-hoa-qua-trong-anh-mau-hay

[2] Facial landmarks with dlib, OpenCV, and Python, https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/ , truy cập ngày 04/06/2022

[3] xuepeng.shi, shiguang.shan, meina.kan, shuzhe.wu, xilin.chen. Real-Time Rotation-Invariant Face Detection with Progressive Calibration Networks.

[4] Anwar Saeed, Ahmed Ghoneim and Ayoub Al-Hamadi. Article in Sensors · September 2015. Head Pose Estimation on Top of Haar-Like Face Detection: A Study Using the Kinect Sensor.

[5] Euclides Arcoverde, George D. C. Cavalcanti and Tsang Ing Ren. Article in Integrated Computer Aided Engineering · April 2014. Enhanced real-time head pose estimation system for mobile device.