



TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO KẾT THÚC HỌC PHẦN
THỊ GIÁC MÁY TÍNH

**PHÁT HIỆN ĐỐI TƯỢNG GẤU MÈO
BẰNG THUẬT TOÁN R-CNN**

Giảng viên hướng dẫn: **PGS.TS Nguyễn Thanh Bình**

Sinh viên thực hiện:

1. Nguyễn Phước An 22DH110271
2. Lương Tiến Đạt 22DH114497
3. Trần Thanh Quang 22DH112007
4. Lê Phạm Minh Quân 22DH112007

Thành phố Hồ Chí Minh, tháng 11 năm 2024



TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO KẾT THÚC HỌC PHẦN
THỊ GIÁC MÁY TÍNH

THUẬT TOÁN R-CNN TRONG PHÁT
HIỆN ĐÓI TƯỢNG GÂU MÈO

Mã lớp học phần: 241123056403

Năm học: 2024 – 2025

Học kỳ: 1

Sinh viên thực hiện:

- Nguyễn Phước An 22DH114428
- Lương Tiến Đạt 22DH114497
- Trần Thanh Quang 22DH114891
- Lê Phạm Minh Quân 22DH114703

Thành phố Hồ Chí Minh, tháng 11 năm 2024

MỤC LỤC

DANH MỤC HÌNH	i
DANH MỤC BẢNG	ii
CHƯƠNG 1. GIỚI THIỆU.....	1
1.1. Giới thiệu đề tài	1
1.2. Nội dung và mục tiêu đề tài	1
1.2.1. <i>Nội dung</i>	1
1.2.2. <i>Mục tiêu</i>	2
1.3. Giới hạn đề tài.....	2
1.4. Cấu trúc báo cáo	3
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	4
2.1. Giới thiệu về CNN (Convolutional Neural Network)	4
2.1.1. <i>Lớp Convolution (Convolutional Layers)</i>	4
2.1.2. <i>Lớp kích hoạt (Activation Layer)</i>	6
2.1.3. <i>Lớp Pooling</i>	6
2.1.4. <i>Lớp Fully Connected (Dense Layers)</i>	7
2.2. Thuật toán tìm kiếm chọn lọc	8
2.3. Độ đo IoU	8
2.4. Thuật toán NMS (non-maximum suppression)	9
2.5. Thuật toán phân lớp SVM (support vector machine)	10
2.6. CNN với mạng MobileNet của Tensorflow	12
2.6.1. <i>Tensorflow là gì</i>	12
2.6.2. <i>Tại sao lại là MobileNet chứ không phải mô hình khác?</i>	12
2.6.3. <i>Thuật toán của MobileNetV2</i>	13
CHƯƠNG 3. PHƯƠNG PHÁP R-CNN TRONG PHÁT HIỆN ĐỐI TƯỢNG	15
3.1. Yêu cầu bài toán	15
3.2. Giải thuật thực hiện	16
3.3. Phương pháp đánh giá	18
CHƯƠNG 4. HIỆN THỰC KẾT QUẢ.....	20
4.1. Yêu cầu hệ thống và tập dữ liệu	20
4.1.1. <i>Yêu cầu hệ thống</i>	20
4.1.2. <i>Tập dữ liệu gấu mèo</i>	22
4.2. Kết quả thực nghiệm.....	22
4.2.1. <i>Tiền xử lý ảnh</i>	22
4.2.2. <i>Train model với MobileNetV2 của Tensorflow</i>	23
4.2.3. <i>Phát hiện đối tượng raccoon với R-CNN</i>	24

CHƯƠNG 5. KẾT LUẬN	27
5.1. Kết quả đạt được.....	27
5.2. Ưu và nhược điểm	27
5.2.1. Ưu điểm	27
5.2.2. Nhược điểm.....	27
5.2.3. Hướng mở rộng trong tương lai	28
TÀI LIỆU THAM KHẢO.....	29
PHỤ LỤC CODE DEMO	30

DANH MỤC HÌNH

Hình 2.1: Quá trình tích chập thông qua bộ lọc.....	5
Hình 2.2: Minh họa quá trình trích xuất đặc trưng (feature maps).....	6
Hình 2.3: phép tính độ đo IoU	8
Hình 2.4: Minh họa quá trình thực hiện NMS.....	9
Hình 2.5: Minh họa giải thuật SVM	10
Hình 2.6: Minh họa khoảng cách trong SVM	11
Hình 2.7: Residual Block.....	13
Hình 2.8: So sánh với mạng neural thông thường với residual	13
Hình 3.1: Mục tiêu đầu vào và đầu ra của mô hình.....	16
Hình 4.1: Minh họa bộ dữ liệu gấu mèo	22
Hình 4.2: Minh họa tập dữ liệu sau khi xử lý	23
Hình 4.3: trực quan hóa số lượng hai lớp	23
Hình 4.4: trực quan hóa kết quả huấn luyện	24
Hình 4.5: Thực nghiệm kết quả lần 1	25
Hình 4.6: Thực nghiệm kết quả lần 2	25
Hình 4.7: Thực nghiệm kết quả lần 3	25
Hình 4.8: Thực nghiệm kết quả lần 4	26

DANH MỤC BẢNG

CHƯƠNG 1. GIỚI THIỆU

1.1. Giới thiệu đề tài

Phát hiện đối tượng là một trong những bài toán cơ bản trong lĩnh vực thị giác máy tính. Mục tiêu của bài toán là xác định các vị trí và phân loại các đối tượng xuất hiện trong ảnh hoặc video. Trong đề tài này, chúng ta sẽ tập trung vào việc phát hiện đối tượng là gấu mèo - một loại động vật có ngoại hình dễ nhận biết, nhưng cũng có nhiều điểm tương đồng với các loài động vật khác như mèo, chó, gấu, khiến việc nhận diện và phân biệt trở nên khó khăn.

Đề tài nghiên cứu "Thuật toán R-CNN trong phát hiện đối tượng gấu mèo" là một ứng dụng cụ thể của thị giác máy tính. Qua nghiên cứu này, chúng ta không chỉ hiểu sâu hơn về phương pháp R-CNN mà còn có thể mở rộng và phát triển các ứng dụng phát hiện đối tượng trong thực tế, ví dụ như giám sát động vật hoang dã, bảo tồn động vật quý hiếm, hay hỗ trợ các nhà khoa học trong nghiên cứu sinh thái.

1.2. Nội dung và mục tiêu đề tài

1.2.1. Nội dung

Đề tài này tập trung vào việc nghiên cứu và ứng dụng thuật toán R-CNN trong bài toán phát hiện và nhận diện đối tượng là gấu mèo trong ảnh và video. Các nội dung chính bao gồm:

- **Tổng quan về thuật toán R-CNN:** Giới thiệu về lịch sử phát triển, kiến trúc và các cải tiến của R-CNN, đồng thời phân tích các đặc điểm nổi bật của nó trong phát hiện đối tượng.
- **Phân tích thuật toán R-CNN trong phát hiện đối tượng:** Trình bày chi tiết các bước của thuật toán R-CNN bao gồm tạo vùng đề xuất, trích xuất đặc trưng và phân loại đối tượng trong vùng, cùng với một số cải tiến như Fast R-CNN và Faster R-CNN nhằm tăng hiệu suất tính toán và độ chính xác.
- **Áp dụng R-CNN trong nhận diện gấu mèo:** Xây dựng và huấn luyện mô hình R-CNN dựa trên tập dữ liệu chứa các hình ảnh của gấu mèo, sau đó thử

nghiệm để đánh giá khả năng phát hiện đối tượng gấu mèo trong môi trường phức tạp và phân biệt chúng với các động vật khác có đặc điểm tương tự.

- **Đánh giá và phân tích kết quả:** Đánh giá hiệu quả mô hình qua các chỉ số như độ chính xác, độ nhạy, và F1-score, cùng với phân tích các yếu tố ảnh hưởng đến hiệu suất của mô hình.

1.2.2. Mục tiêu

Mục tiêu của đề tài bao gồm các khía cạnh nghiên cứu và ứng dụng như sau:

- **Nghiên cứu thuật toán R-CNN và các cải tiến:** Hiểu rõ cơ chế hoạt động của R-CNN trong việc phát hiện đối tượng, bao gồm cả cách tạo vùng đề xuất và phương pháp trích xuất đặc trưng để áp dụng cho các bài toán nhận diện trong thực tế.
- **Xây dựng mô hình phát hiện gấu mèo:** Phát triển một mô hình dựa trên R-CNN để có thể phát hiện chính xác và nhanh chóng đối tượng là gấu mèo trong các bức ảnh hoặc video có độ phức tạp cao.
- **Đánh giá hiệu quả phát hiện gấu mèo:** Đo lường hiệu suất của mô hình qua các chỉ số, từ đó đánh giá khả năng áp dụng trong thực tế và khả năng mở rộng để phát hiện các động vật khác.
- **Đề xuất cải tiến cho mô hình R-CNN trong phát hiện gấu mèo:** Thủ nghiệm và điều chỉnh các tham số, kiến trúc mạng, hoặc kết hợp với các thuật toán khác nhằm tăng độ chính xác và tốc độ nhận diện.

1.3. Giới hạn đề tài

Đề tài này có một số giới hạn nhất định do đặc điểm của thuật toán R-CNN và các yếu tố ảnh hưởng đến việc phát hiện đối tượng trong các hình ảnh thực tế. Cụ thể:

- **Giới hạn về dữ liệu:** Số lượng và chất lượng hình ảnh chưa đủ (200 ảnh) làm ảnh hưởng đến độ chính xác của mô hình R-CNN, tập dữ liệu không đủ lớn hoặc không đủ đa mô hình sẽ khó nhận diện chính xác trong các tình huống thực tế. Hình ảnh bị nhiễu hoặc hình ảnh chất lượng thấp có thể làm giảm hiệu suất phát hiện của R-CNN.

- **Hiệu suất tính toán:** R-CNN gốc có quá trình tính toán khá chậm do phải tạo và xử lý nhiều vùng đề xuất, khiến nó chưa thực sự phù hợp để áp dụng trong các hệ thống yêu cầu phát hiện đối tượng thời gian thực.
- **Giới hạn về độ chính xác:** Thuật toán R-CNN gặp khó khăn trong việc phân biệt gấu mèo với các động vật khác có hình dáng hoặc màu sắc tương tự. Điều này đặc biệt khó khăn khi các đối tượng tương tự xuất hiện trong cùng một bức ảnh.

1.4. Cấu trúc báo cáo

- 1) **Phản giới thiệu:** Giới thiệu về đề tài, lý do chọn đề tài và mục tiêu nghiên cứu của mô hình.
- 2) **Cơ sở lý thuyết:** Giới thiệu tổng quan về các thuật toán, mô hình liên quan tới đề tài.
- 3) **Phương pháp nghiên cứu:** Giới thiệu tổng quan về yêu cầu của bài toán (input, output của thuật toán), mô tả quá trình thực hiện của thuật toán, giới thiệu về phương pháp đánh giá thuật toán.
- 4) **Hiện thực kết quả:** Giới thiệu về môi trường, ngôn ngữ, các thư viện, tập dữ liệu để thực hiện thuật toán sau đó thực nghiệm kết quả, đánh giá và đưa ra nhận xét.
- 5) **Kết luận:** Tổng kết kết quả chính đạt được trong đề tài và nêu ra ưu nhược điểm, hướng phát triển và đề xuất các hướng nghiên cứu tiếp theo để cải thiện mô hình.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về CNN (Convolutional Neural Network)

CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo, đặc biệt hiệu quả trong việc xử lý dữ liệu dạng lưới như hình ảnh. CNN được thiết kế để học và nhận diện các đặc trưng trong hình ảnh bằng cách tự động phát hiện các mẫu (patterns) qua nhiều lớp tích chập (convolutional layers). Thuật toán CNN thường được sử dụng trong các bài toán thị giác máy tính như phân loại hình ảnh, nhận diện đối tượng, và phát hiện khuôn mặt.

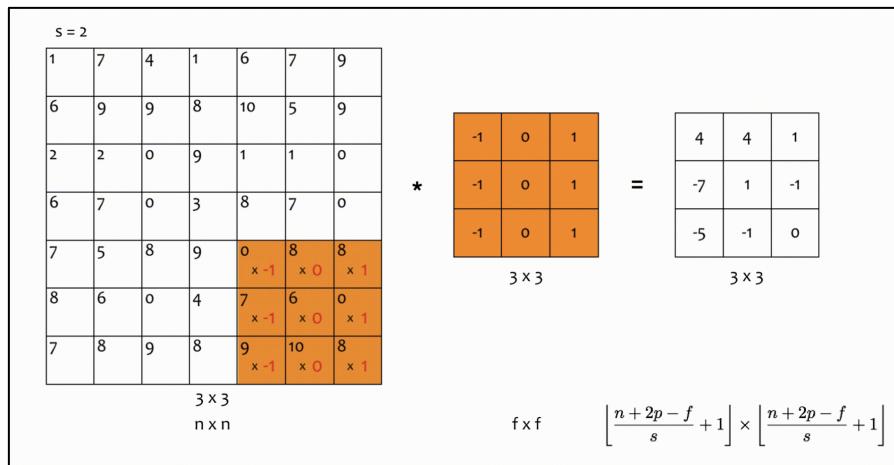
Cấu trúc cơ bản của CNN bao gồm 4 lớp:

- Lớp Convolutional (Convolutional Layers)
- Lớp Phi Tuyến (Activation Layer)
- Lớp Pooling (Pooling Layers)
- Lớp Fully Connected (Dense Layers)

2.1.1. Lớp Convolution (Convolutional Layers)

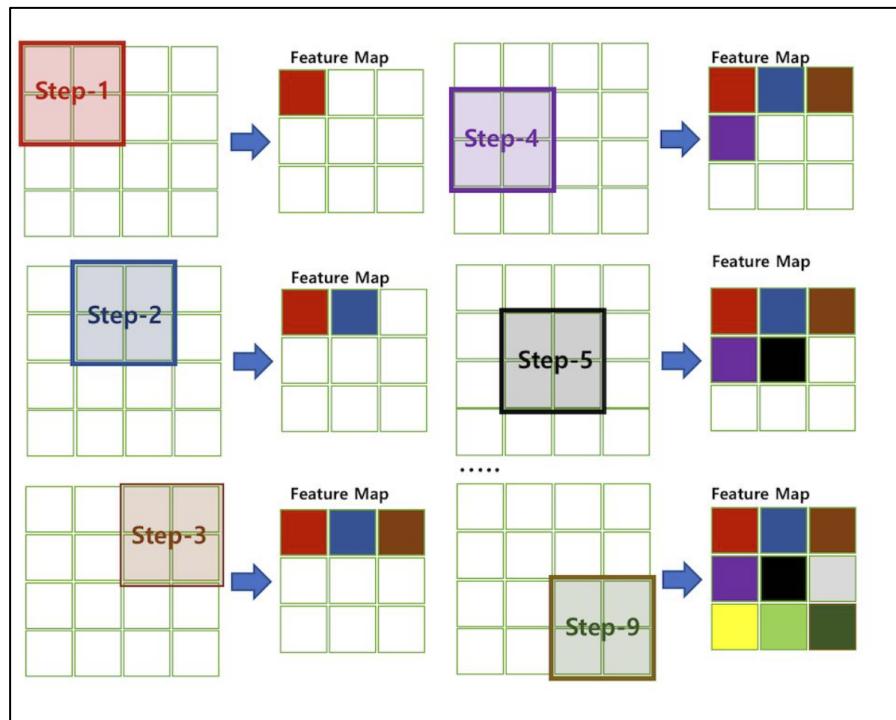
Lớp Convolutional (lớp tích chập) là thành phần cốt lõi trong một mạng CNN, chuyên dùng để trích xuất các đặc trưng. Lớp convolutional có cách hoạt động và các tham số chính như sau:

- 1) **Bộ lọc (Filter):** Là một ma trận nhỏ (thường có kích thước như 3x3, 5x5), chứa các giá trị trọng số. Quá trình tích chập là việc trượt bộ lọc qua các vùng của ảnh. Tại mỗi vị trí, bộ lọc sẽ thực hiện phép nhân và cộng giữa các giá trị trong bộ lọc và các giá trị trong vùng ảnh đó => Kết quả là một giá trị đặc trưng mới, được gọi là feature map.



Hình 2.1: Quá trình tích chập thông qua bộ lọc

- 2) **Stride:** Là khoảng cách mà bộ lọc di chuyển trên ảnh đầu vào sau mỗi bước tính toán. Stride thường là 1 hoặc 2. Nếu stride là 1, bộ lọc di chuyển từng ô một. Nếu stride là 2, bộ lọc sẽ bỏ qua một ô giữa mỗi lần trượt, giúp giảm kích thước của map đặc trưng.
- 3) **Padding:** Khi bộ lọc trượt qua biên của hình ảnh, có thể xuất hiện tình huống vùng trượt không phủ kín ảnh. Để giải quyết, CNN sử dụng padding để thêm các hàng hoặc cột giá trị 0 quanh biên ảnh, giúp bảo toàn kích thước của ảnh đầu ra. Các loại padding phổ biến:
 - Valid Padding
 - Same Padding
- 4) **Feature map:** Sau khi tích chập, kết quả là các map đặc trưng (feature maps) thể hiện các đặc trưng trích xuất từ ảnh gốc. Mỗi bộ lọc tạo ra một feature map riêng. Các map đặc trưng này là đầu vào cho lớp kế tiếp.



Hình 2.2: Minh họa quá trình trích xuất đặc trưng (feature maps)

2.1.2. Lớp kích hoạt (Activation Layer)

Lớp kích hoạt (Activation Layer) là một lớp giúp tạo tính phi tuyến cho mạng bằng cách áp dụng các hàm kích hoạt (activation functions), lớp này giúp mô hình có khả năng học các mối quan hệ phức tạp và phi tuyến tính giữa các đặc trưng. Một số hàm kích hoạt phổ biến:

- Hàm ReLU
- Hàm Leaky ReLU
- Hàm Sigmoid
- Hàm Tanh
- Hàm Softmax

2.1.3. Lớp Pooling

Lớp pooling là một lớp trong CNN có nhiệm vụ giảm kích thước của các map đặc trưng (feature maps) đầu ra từ lớp convolutional. Quá trình này giúp giảm số lượng tham số và tính toán trong mạng, đồng thời tăng cường khả năng khái quát của mô hình bằng cách giữ lại các đặc trưng quan trọng trong khi loại bỏ thông tin thừa, ngoài ra giúp cho mô hình giảm nguy cơ overfitting và duy trì tính bất biến về vị trí.

Có một số loại pooling phổ biến, tùy thuộc vào cách chọn giá trị từ vùng con của feature map, sau đây là các phương pháp Pooling thường dùng:

- Max Pooling: Chọn giá trị lớn nhất giúp giữ lại các đặc trưng mạnh nhất.
- Average Pooling: Tính giá trị trung bình có thể làm mất đi các đặc trưng quan trọng trong map đặc trưng.
- Global Pooling: Áp dụng Max Pooling hoặc Average Pooling trên toàn bộ feature map, tạo ra một giá trị duy nhất.

2.1.4. Lớp Fully Connected (Dense Layers)

Lớp Fully Connected, còn gọi là lớp Dense là lớp cuối cùng trong hầu hết các mạng CNN và có nhiệm vụ thực hiện quá trình phân loại hoặc hồi quy dựa trên các đặc trưng trích xuất từ các lớp trước đó. Lớp này kết nối tất cả các nơ-ron của lớp trước đó với mỗi nơ-ron trong lớp hiện tại, tạo thành một mạng nơ-ron dày đặc

Vai trò của Lớp Fully Connected trong CNN:

- Phân loại ảnh vào các nhãn đầu ra
- Kết hợp đặc trưng để tạo ra một bức tranh toàn diện của dữ liệu.
- Chuyển đổi thành đầu ra cuối cùng

Cấu Trúc của Lớp Fully Connected:

- Mỗi nơ-ron trong lớp fully connected kết nối với tất cả các nơ-ron của lớp trước đó.
- Mỗi kết nối này có một trọng số riêng và một độ chêch (bias), được học trong quá trình huấn luyện.
- Công thức tính cho mỗi nơ-ron y (z) trong lớp fully connected là:

$$z = \sum_{i=1}^N w_i * x_i + b$$

Trong đó:

- a. w_i : trọng số kết nối giữa nơ-ron đầu vào x_i và nơ-ron hiện tại y (z)
- b. b : là bias (tham số tự do)

Các Hàm Kích Hoạt Phổ Biến cho Lớp Fully Connected:

- Hàm ReLU (Rectified Linear Unit)
- Hàm Sigmoid
- Hàm Softmax

2.2. Thuật toán tìm kiếm chọn lọc

Thuật toán tìm kiếm chọn lọc (selection search) lần đầu tiên được tác giả uilling và các cộng sự công bố trong một bài báo được phát hành vào năm 2013.

Thuật toán tìm kiếm chọn lọc là một phương pháp rất phổ biến trong lĩnh vực thi giác máy tính đặc biệt là trong các bài toán trích xuất vùng đề xuất (region proposal) cho các hệ thống phát hiện đối tượng (object detection).

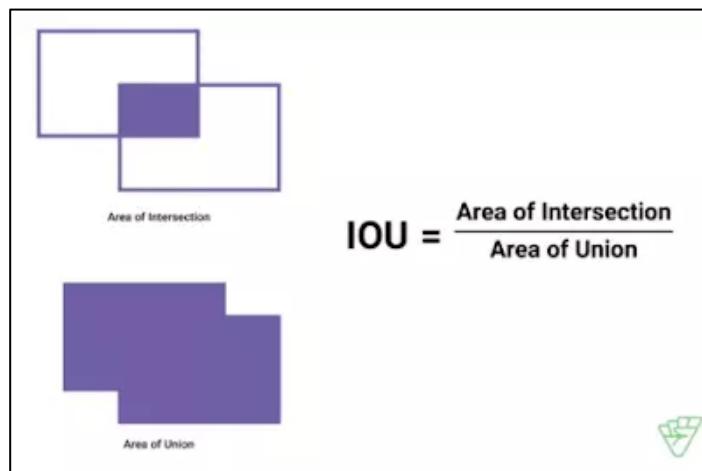
Để có thể xuất ra các vùng đề xuất như thế thì người ta dùng phương pháp đó là phân đoạn ảnh dựa trên nhiều tiêu chí như là sự tương đồng về màu sắc, kết cấu, kích thước, hình dạng.

Sau đó ta sẽ trực quan hóa vùng đề xuất đó bằng các bounding box bao xung quanh thường là hình chữ nhật hoặc hình vuông và được giới hạn bởi điểm trên cùng bên trái (x_1, y_1) và điểm dưới cùng bên phải (x_2, y_2).

2.3. Độ đo IoU

Độ đo IoU là phép đo để đánh giá độ chính xác trong phát hiện đối tượng dựa trên phần giao nhau và phần hợp nhau của khung giới hạn thực và khung giới hạn dự đoán.

Được tính bằng công thức:



Hình 2.3: phép tính độ đo IoU

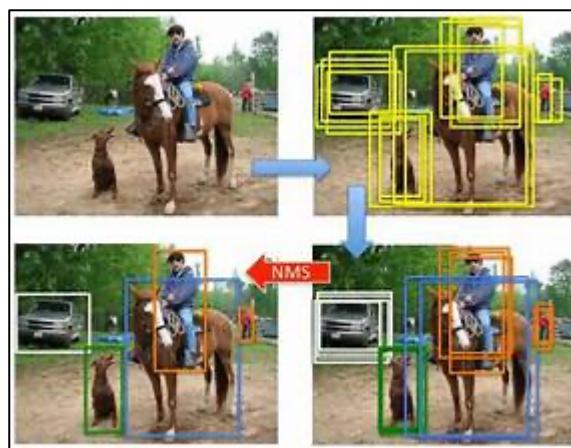
Nếu giá trị này lớn hơn 0.5 thì mô hình được coi là dự đoán tốt. Ngưỡng này có thể thay đổi tùy theo yêu cầu của bài toán nhưng cao lắm cũng chỉ được 0.8, 0.9.

Ví dụ về cách tính độ đo IoU:

- Giả sử có hai hình chữ nhật, hình chữ nhật A có tọa độ góc trên cùng bên trái là (1, 1), tọa độ góc dưới cùng bên phải là (4, 4), hình chữ nhật B có tọa độ góc trên cùng bên trái là (2, 2), tọa độ góc dưới cùng bên phải là (5, 5)
- $\text{Area of Intersection} = (4-2) * (4-2)$
 $= 4$
- $\text{Area of Union} = (4-1) * (4-1) + (5-2) * (5-2) - \text{Area of Intersection}$
 $= 3*3 + 3*3 - 4$
 $= 14$
- $\text{IoU} = \text{Area of Intersection} / \text{Area of Union}$
 $= 0.286$

2.4. Thuật toán NMS (non-maximum suppression)

Thuật toán NMS là một kỹ thuật thường được sử dụng trong các mô hình phát hiện đối tượng (object detection) để loại bỏ các bounding box dư thừa, chỉ giữ lại những box có độ tin cậy cao nhất cho mỗi đối tượng. Nghĩa là sau khi ta áp dụng thuật toán tìm kiếm chọn lọc (selective search) thì mô hình sẽ trả về ảnh có chứa nhiều bounding box có thể đúng có thể sai, chồng chéo lên nhau thì NMS có nhiệm vụ loại bỏ những bounding box dư thừa đó và giữ lại bounding box chính xác nhất dựa vào độ đo IoU.



Hình 2.4: Minh họa quá trình thực hiện NMS

Với đầu vào là một tập hợp các bounding box với confidence scores tương ứng. Các bước thực hiện của thuật toán NMS như sau:

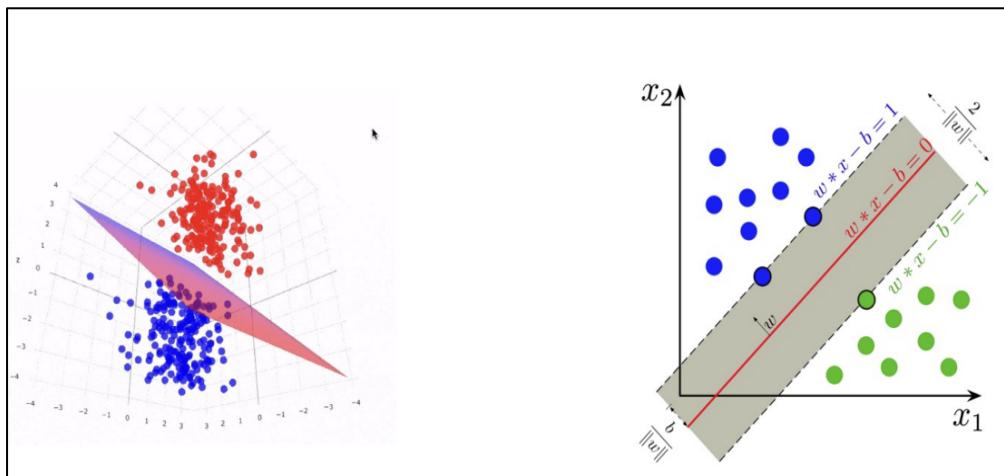
- Bước 1: Sắp xếp các bounding box theo giá trị confidence scores giảm dần sau đó lưu vào tập kết quả.
- Bước 2: Chọn bounding box có confidence scores cao nhất.
- Bước 3: Tính độ đo IoU giữa bounding box đó với các bounding box còn lại. Sau đó loại bỏ các bounding box có IoU lớn hơn một ngưỡng xác định thường là 0,5.
- Bước 4: Lặp lại bước 2 và bước 3 cho tới khi không còn bounding box nào trong tập ban đầu.

Ưu điểm: Giảm thiểu số lượng bounding box dư thừa, giúp mô hình phát hiện đối tượng chính xác hơn.

Nhược điểm: Hiệu quả của thuật toán NMS phụ thuộc vào ngưỡng IoU được chọn. Nếu ngưỡng này quá cao hoặc quá thấp, có thể dẫn tới việc loại bỏ nhầm hoặc giữ lại quá nhiều bounding box.

2.5. Thuật toán phân lớp SVM (support vector machine)

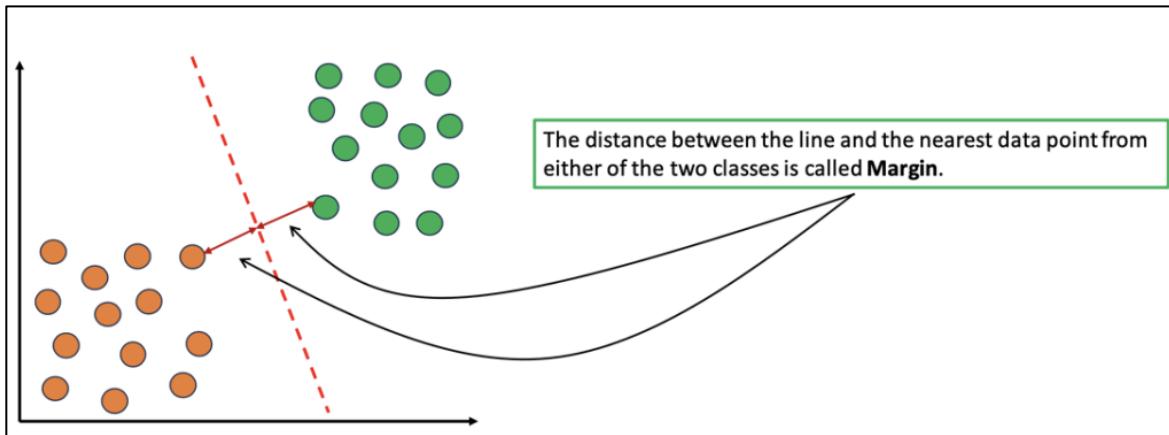
Thuật toán SVM (Support Vector Machine) là một phương pháp học máy được sử dụng chủ yếu cho các bài toán phân loại và hồi quy. Mục tiêu của SVM là tìm ra một "siêu phẳng" (hyperplane) trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng.



Hình 2.5: Minh họa giải thuật SVM

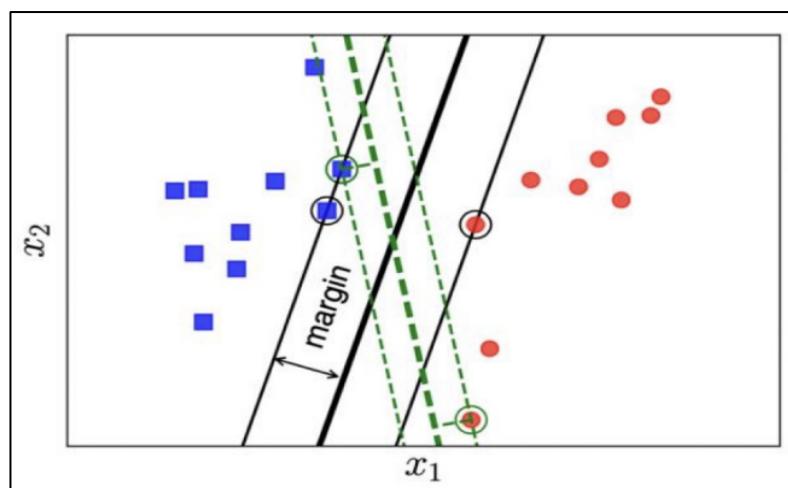
- Độ rộng của margin:

- Nếu ta định nghĩa độ rộng thõa mãn của một lớp tỉ lệ thuận với khoảng cách gần nhất từ một điểm của lớp đó tới đường/mặt phân chia, thì lớp tròn đỏ sẽ không thõa mãn vì đường phân chia gần nó hơn lớp vuông xanh là rất nhiều.
- Chúng cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi lớp (các điểm được khoanh tròn) tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là margin.



Hình 2.6: Minh họa khoảng cách trong SVM

- Việc margin rộng hơn sẽ mang lại hiệu quả phân lớp tốt hơn vì sự phân chia giữa hai lớp là rạch ròi hơn.
- Bài toán tối ưu trong SVM chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất.



- **Soft Margin và Hard Margin:** Trong trường hợp dữ liệu không hoàn toàn phân chia được (có chồng lấn), SVM sẽ dùng phương pháp "Soft Margin" để cho phép một số điểm dữ liệu vi phạm điều kiện phân chia.
- **Hạt nhân (Kernel):** Để xử lý các bài toán phi tuyến, SVM có thể dùng hàm kernel để ánh xạ dữ liệu vào không gian có số chiều cao hơn, nơi các lớp dữ liệu có thể phân chia tuyến tính. Một số kernel phổ biến là:
 - **Linear Kernel:** Sử dụng khi dữ liệu có thể phân chia tuyến tính.
 - **Polynomial Kernel:** Tăng sức mạnh biểu diễn bằng cách dùng các hàm đa thức.
 - **RBF (Radial Basis Function) Kernel:** Thường dùng trong các bài toán phi tuyến.
- **Ưu điểm của SVM:**
 - Hiệu quả với dữ liệu phân biệt rõ ràng.
 - Khả năng chống quá khớp tốt khi số đặc trưng lớn.
 - Hỗ trợ phân loại phi tuyến nhờ các hàm kernel.
- **Nhược điểm của SVM:**
 - Thời gian tính toán cao với dữ liệu lớn.
 - Hiệu suất giảm khi có nhiều nhiễu hoặc dữ liệu chồng lấn cao.

2.6. CNN với mạng MobileNet của Tensorflow

2.6.1. Tensorflow là gì

Tensorflow là một thư viện mã nguồn mở phổ biến do Google phát triển để xây dựng học máy và học sâu và triển khai chúng trên nhiều nền tảng như máy tính, thiết bị di động và thiết bị IoT. TensorFlow đặc biệt mạnh trong xử lý các bài toán học sâu, các kiến trúc mạng phức tạp như Mạng Nơ-ron Tích chập (CNN).

2.6.2. Tại sao lại là MobileNet chứ không phải mô hình khác?

MobileNet là một mô hình nhẹ và tối ưu hóa của CNN, được thiết kế cho các ứng dụng yêu cầu độ chính xác cao mà vẫn phải chạy hiệu quả trên các thiết bị có tài nguyên hạn chế, như điện thoại thông minh. Trong thời đại xung quanh mọi người đều sử dụng thiết bị di động thì việc lựa chọn và sử dụng mô hình vào nó là cần thiết.

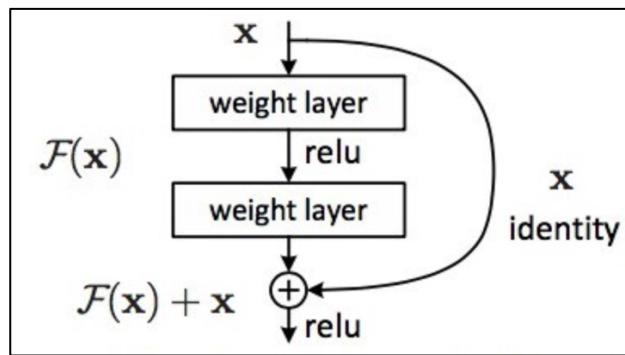
Trong khi các mạng lớn hơn như VGG16 hay ResNet có hiệu suất cao, chúng lại tiêu tốn nhiều tài nguyên, như bộ nhớ và khả năng tính toán, điều này gây khó khăn khi triển khai trên các thiết bị di động.

2.6.3. Thuật toán của MobileNetV2

MobileNetV2 có một số điểm cải tiến so với MobileNetV1 là số lượng tham số và số lượng các phép tính ít hơn vì thế giúp cho nó có độ chính xác cao hơn trong khi vẫn giữ được các đặc trưng cần thiết.

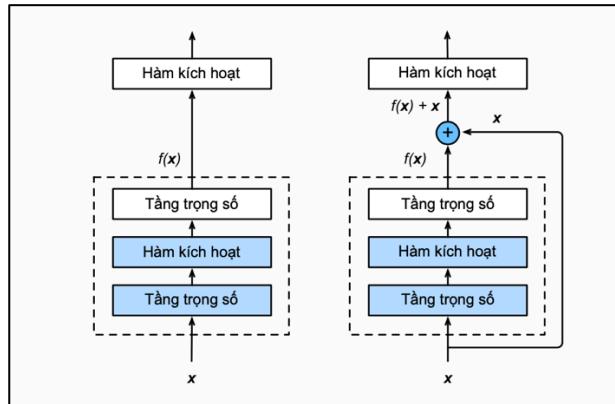
Điểm cốt lõi của MobileNetV2 là việc sử dụng các khối tích chập có tên là Inverted Residuals và Linear Bottlenecks:

1) Inverted Residual Block



Hình 2.7: Residual Block

MobileNetV2 sử dụng những kết nối tắt giống như mạng ResNet giúp giữ thông tin không bị mất bằng cách đưa các khối ở layer trước được cộng trực tiếp vào layer liền sau. Kết nối được điều chỉnh sao cho số kênh ở input và output của mỗi block residual được thắt hẹp lại.



Hình 2.8: So sánh với mạng neural thông thường với residual

2) Linear Bottlenecks

Việc sử dụng các biến đổi phi tuyến (như biến đổi qua ReLu hoặc sigmoid) tại input và output của các residual block sẽ làm cho thông tin bị mất mát. Vì thế mỗi khối của MobileNetV2 kết thúc bằng một lớp tuyến tính gọi là linear bottleneck. Vì ReLU có thể làm mất thông tin khi áp dụng cho các đầu ra có giá trị nhỏ, dẫn đến việc mất các đặc trưng quan trọng. Sử dụng linear bottleneck giúp giữ lại nhiều thông tin hơn và cải thiện khả năng của mô hình trong việc học các đặc trưng chi tiết của hình ảnh.

CHƯƠNG 3. PHƯƠNG PHÁP R-CNN TRONG PHÁT HIỆN ĐỐI TƯỢNG

3.1. Yêu cầu bài toán

R-CNN (Regions with Convolutional Neural Network Features) là một trong những phương pháp tiên phong trong nhận diện và phát hiện đối tượng. Mục tiêu của bài toán là xác định và vẽ các hộp giới hạn (bounding box) xung quanh các đối tượng cụ thể, ở đây là racoon (gấu mèo), trong một ảnh đầu vào.

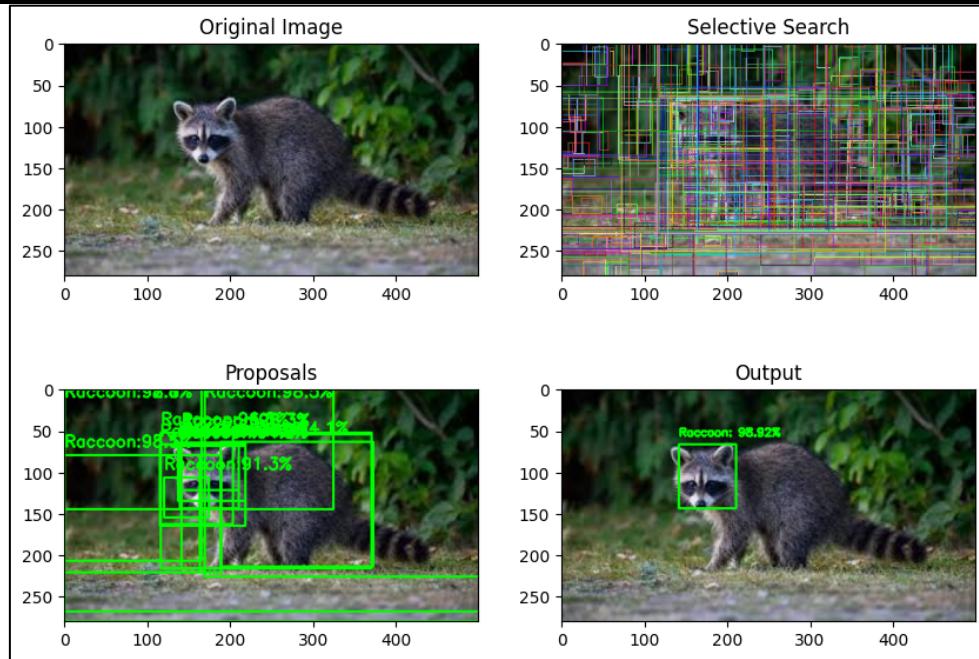
Các bước chính cần thực hiện trong bài toán phát hiện đối tượng với R-CNN:

- Đầu vào: Một hình ảnh chứa các đối tượng tiềm năng cần phát hiện.
- Đầu ra: Vị trí của các đối tượng được phát hiện dưới dạng hộp giới hạn, với nhãn và xác suất dự đoán.

Yêu cầu:

1. Xác định vùng đề xuất: Áp dụng phương pháp Selective Search để tạo các vùng đề xuất cho các khu vực tiềm năng chứa đối tượng trong ảnh.
2. Phân loại vùng: Đưa các vùng đề xuất vào mạng CNN đã huấn luyện để xác định xác suất mỗi vùng chứa đối tượng mong muốn.

Xử lý Non-Maximum Suppression (NMS): Loại bỏ các hộp chòng chéo có xác suất thấp, chỉ giữ lại hộp có độ tin cậy cao nhất cho mỗi đối tượng.



Hình 3.1: Mục tiêu đầu vào và đầu ra của mô hình

3.2. Giải thuật thực hiện

Để giải quyết bài toán, thuật toán R-CNN trong phát hiện đối tượng gấu mèo ta tiến hành theo hai phần sau bao gồm các bước sau:

- 1) Xử lý dữ liệu và huấn luyện mô hình

Bước 1: Chuẩn Bị Dữ Liệu

- Đọc và xử lý dữ liệu nhãn: Đọc file CSV chứa thông tin về tọa độ bounding box của raccoon trong các ảnh.
- Vẽ hộp giới hạn (Bounding Box): Tạo hàm để vẽ hộp giới hạn lên ảnh dựa trên thông tin nhãn.
- Chọn vùng chứa đối tượng: Lọc thông tin bounding box từ dữ liệu nhãn dựa trên tên ảnh.
- Tải ảnh: Đọc ảnh từ đường dẫn và vẽ hộp giới hạn lên ảnh.

Bước 2: Tạo và Lưu Các Vùng Đề Xuất (ROI)

- Khởi tạo công cụ Selective Search: Sử dụng thư viện OpenCV để tạo đối tượng Selective Search và xác định các vùng đề xuất (ROI).

- Lọc ROI tích cực và tiêu cực: Duyệt qua các ROI và tính toán chỉ số IoU với các bounding box chuẩn, xác định ROI nào tích cực ($\text{IoU} > 0.7$) và ROI nào tiêu cực ($\text{IoU} < 0.05$).
- Lưu ROI tích cực và tiêu cực: Lưu các ROI vào thư mục tương ứng (POSITIVE_PATH cho tích cực, NEGATIVE_PATH cho tiêu cực).

Bước 3: Tiền Xử Lý Dữ Liệu Đầu Vào

- Khởi tạo tham số huấn luyện: Cài đặt các tham số như tốc độ học, số vòng lặp huấn luyện, và kích thước batch.
- Tiền xử lý ảnh và nhãn: Đọc và tiền xử lý các ảnh, thay đổi kích thước, mã hóa nhãn bằng One-Hot Encoding.
- Chia dữ liệu: Chia dữ liệu thành tập huấn luyện và kiểm tra (80% huấn luyện, 20% kiểm tra).
- Áp dụng kỹ thuật Augmentation: Sử dụng ImageDataGenerator để tăng cường dữ liệu huấn luyện.

Bước 4: Xây Dựng và Huấn Luyện Mô Hình

- Sử dụng MobileNetV2 làm mô hình cơ sở: Chọn MobileNetV2 làm mô hình cơ sở và thay thế lớp fully connected cuối cùng.
- Biên dịch mô hình: Sử dụng hàm mất mát binary_crossentropy, bộ tối ưu Adam, và theo dõi độ chính xác.
- Huấn luyện mô hình: Dùng các kỹ thuật augmentation để huấn luyện mô hình trong một số vòng lặp nhất định.

Bước 5: Đánh Giá Mô Hình và Lưu Trữ

- Đánh giá mô hình: Dự đoán nhãn cho tập kiểm tra và in ra báo cáo phân loại.
- Vẽ confusion matrix: Trực quan hóa hiệu suất mô hình bằng confusion matrix.
- Trực quan hóa quá trình huấn luyện: Vẽ đồ thị Loss và Accuracy qua các epoch.

- Lưu mô hình và encoder: Lưu mô hình đã huấn luyện và encoder nhãn vào file để sử dụng sau này.

2) Thực nghiệm kết quả và đưa ra đánh giá cải tiến mô hình

Bước 1: Tải Mô Hình và Bộ Mã Hóa Nhãn:

- Sử dụng load_model() để tải mô hình đã được huấn luyện cho tác vụ phát hiện raccoon. Dùng pickle.loads() để tải bộ mã hóa nhãn (label encoder) đã lưu trước đó.

Bước 2: Tải và áp dụng tìm kiếm có chọn lọc cho ảnh đầu vào:

- Đọc ảnh đầu vào từ thư mục và hiển thị với OpenCV và cv2_imshow().
- Tạo Vùng Đề Xuất với Selective Search:
- Áp dụng kỹ thuật Selective Search (createSelectiveSearchSegmentation) để xác định các vùng đề xuất trong ảnh.

Bước 3: Tiền xử lý các vùng đề xuất:

- Lấy từng vùng ROI từ ảnh gốc, chuyển đổi từ hệ màu BGR sang RGB, thay đổi kích thước về chuẩn 224x224, và tiến hành tiền xử lý để đưa vào mạng CNN.

Bước 4: Dự đoán lớp đối tượng trên vùng đề xuất

- Sử dụng mô hình đã huấn luyện để dự đoán lớp cho từng vùng đề xuất.
- Lọc ra các vùng có xác suất dự đoán lớn hơn MIN_PROBA và có khả năng là raccoon.

Bước 5: Hiển thị vùng đề xuất có đối tượng

- Áp dụng kỹ thuật NMS để loại bỏ các bounding box trùng lặp, sử dụng ngưỡng overlapThresh để giữ lại các bounding box quan trọng nhất.
- Vẽ các bounding box cuối cùng đã qua xử lý NMS lên ảnh gốc.

3.3. Phương pháp đánh giá

Hiệu quả của R-CNN trong bài toán phát hiện đối tượng gấu mèo được đánh giá bằng các tiêu chí phổ biến như sau:

-
- **Tỷ lệ chèong lán IOU (Intersection over Union)** Tiêu chí IOU đo lường mức độ trùng khớp giữa vùng dự đoán và vùng thực tế có đối tượng. Giá trị IOU trong mô hình có giá trị trên 94%.
 - **Thời gian xử lý** Để đảm bảo thuật toán có thể áp dụng trong thực tế, thời gian phát hiện trên mỗi hình ảnh cũng là một tiêu chí quan trọng. Trong mô hình phát hiện gấu mèo của chúng tôi mỗi bức hình cần trung bình khoảng thời gian là 10s để đưa ra dự đoán cho 200 vùng đề xuất.

CHƯƠNG 4. HIỆN THỰC KẾT QUẢ

4.1. Yêu cầu hệ thống và tập dữ liệu.

4.1.1. Yêu cầu hệ thống

1) Python

Python là một ngôn ngữ thông dịch, điều này nghĩa là ngôn ngữ này trực tiếp chạy từng dòng mã. Sử dụng từ ngữ giống trong tiếng Anh gần gũi với ngôn ngữ con người hơn các ngôn ngữ lập trình khác. Không giống như các ngôn ngữ lập trình khác. Python là một ngôn ngữ cấp cao. Do đó, các lập trình viên không cần phải lo lắng về những chức năng cơ bản của nó như kiến trúc và quản lý bộ nhớ.

Trong hệ thống của chúng tôi sử dụng phiên bản Python 3.10.12.



2) Google Colaboratory

Colaboratory hay còn gọi là Google Colab, là một sản phẩm từ Google Research, nó cho phép thực thi Python trên nền tảng đám mây. Google Colab cho phép tạo và chạy các Jupyter Notebooks trực tuyến mà không cần cài đặt môi trường phát triển phức tạp trên máy tính cá nhân. Tương tự như Jupyter Notebook truyền thống với các cell cho phép thực thi mã Python hoặc viết markdown để tạo nội dung hướng dẫn. Ngoài ra cùng với thuật toán đề xuất code cực kì nhanh chóng và chính xác.



Google Colab cung cấp truy cập miễn phí đến GPU và TPU, đặc biệt hữu ích cho các tác vụ tính toán nặng về mặt số học, đặc biệt là trong lĩnh vực học máy và deep learning.

3) Các thư viện cần thiết

Matplotlib: Các nhà phát triển sử dụng Matplotlib để hiển thị dữ liệu dưới dạng đồ họa hai và ba chiều (2D và 3D) chất lượng cao. Thư viện này thường được sử dụng trong các ứng dụng khoa học

Seaborn: Thư viện vẽ biểu đồ và trực quan hóa dữ liệu, mở rộng matplotlib với các biểu đồ trực quan và dễ sử dụng. Trong thị giác máy tính, nó hữu ích cho việc phân tích và biểu diễn dữ liệu.

Pandas: cung cấp cấu trúc dữ liệu được tối ưu hóa và linh hoạt mà có thể sử dụng để thao tác với dữ liệu chuỗi thời gian và dữ liệu có cấu trúc, chẳng hạn như bảng và nhóm.

NumPy: là một thư viện phổ biến mà các nhà phát triển sử dụng để dễ dàng tạo và quản lý nhóm, thao tác với các hình dạng logic và thực hiện các phép toán đại số tuyến tính.

OpenCV: Là một thư viện mà các nhà phát triển sử dụng để xử lý hình ảnh cho các ứng dụng thị giác máy tính. Thư viện này cung cấp nhiều hàm cho các tác vụ xử lý hình ảnh như đọc và ghi hình ảnh cùng lúc, xây dựng môi trường 3D từ môi trường 2D cũng như chụp và phân tích hình ảnh từ video.

Keras: Là thư viện mạng nơ-ron chuyên sâu của Python với khả năng hỗ trợ tuyệt vời cho việc xử lý dữ liệu, trực quan hóa và hơn thế nữa. Keras hỗ trợ nhiều mạng nơ-ron. Thư viện này có cấu trúc mô-đun mang lại sự linh hoạt cho việc lập trình các ứng dụng sáng tạo.

Tensorflow: là một thư viện mã nguồn mở cung cấp khả năng xử lý tính toán số học dựa trên biểu đồ mô tả sự thay đổi của dữ liệu, trong đó các node là các phép tính toán học còn các cạnh biểu thị luồng dữ liệu.

imutils: Thư viện hỗ trợ các thao tác xử lý ảnh cơ bản như xoay, thay đổi kích thước, dịch chuyển ảnh, giúp đơn giản hóa các thao tác thường dùng trong OpenCV.

sklearn: Thư viện học máy cung cấp các công cụ để xây dựng và đánh giá các mô hình, bao gồm phân loại, hồi quy, phân cụm và các thuật toán học máy, thường được dùng cho xử lý dữ liệu và đánh giá mô hình trong thị giác máy tính.

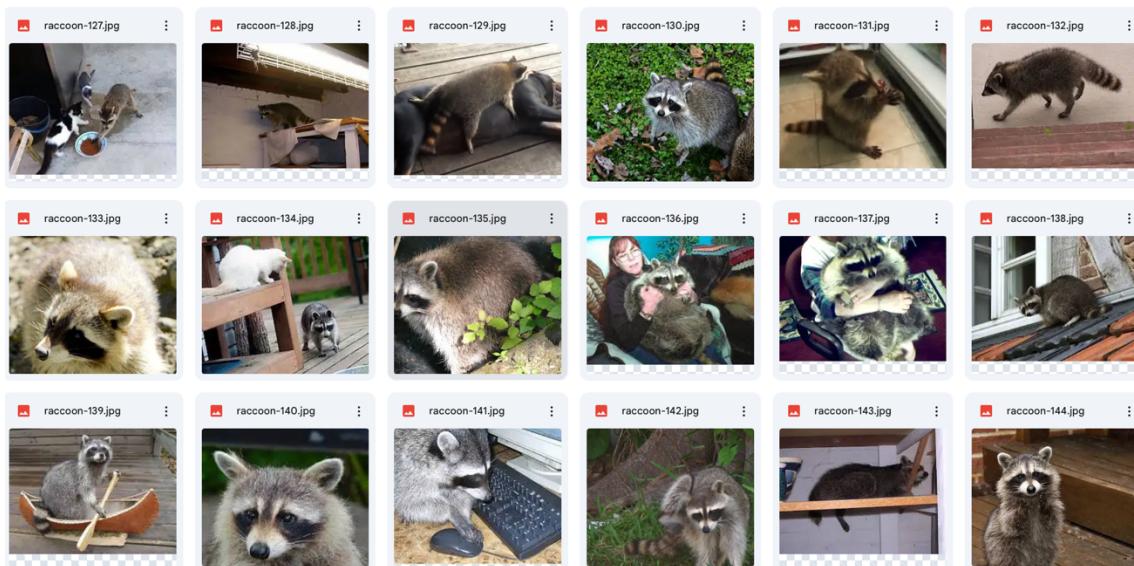
pickle: Thư viện chuẩn để lưu trữ và tải lại dữ liệu và mô hình trong Python, giúp tuần tự hóa (serialization) dữ liệu như mô hình học máy, từ điển, danh sách, v.v.

random: Thư viện chuẩn để tạo số ngẫu nhiên và thực hiện các phép toán ngẫu nhiên, được dùng để chia tập dữ liệu, chọn mẫu ngẫu nhiên hoặc tạo dữ liệu giả.

Image (Pillow): Module từ thư viện Pillow (PIL) hỗ trợ mở, xử lý, và lưu ảnh. Nó có thể đọc và ghi nhiều định dạng ảnh, cung cấp các công cụ để chỉnh sửa và biến đổi ảnh dễ dàng.

4.1.2. Tập dữ liệu gấu mèo

Bộ dữ liệu gấu mèo được thu thập và quản lý bởi Data scientist [Dat Tran](#). Bộ dữ liệu hình ảnh này chứa 200 đối tượng gấu mèo cùng với một file csv chưa tọa độ hộp giới hạn của chúng.

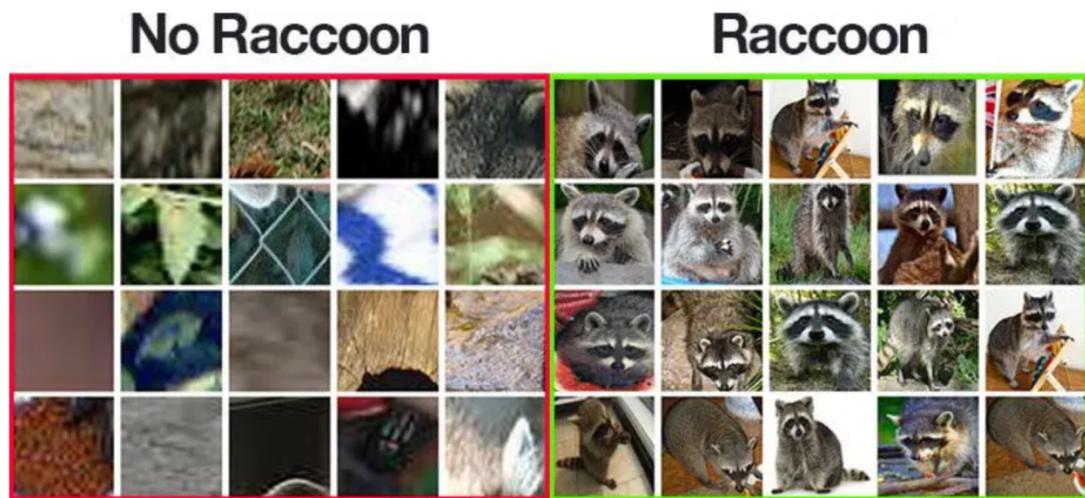


Hình 4.1: Minh họa bộ dữ liệu gấu mèo

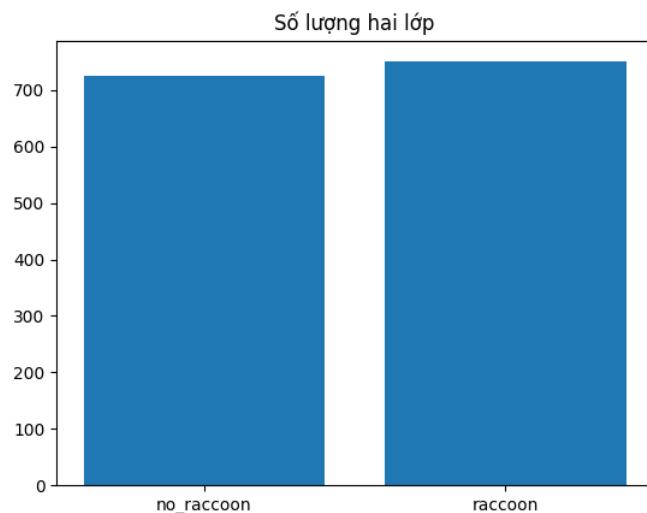
4.2. Kết quả thực nghiệm

4.2.1. Tiền xử lý ảnh

Trước tiên, chúng ta phải xử lý ảnh trong 200 bức ảnh về gấu mèo để tìm ra các vùng tích cực (có độ IoU > 0.7) và các vùng tiêu cực (tìm từ các vùng không phải tích cực và có độ IoU >= 0.05). Mỗi bức hình trong mỗi vùng được lưu không quá 30 hình. Sau khi xử lý ta sẽ có hai folder chia làm hai lớp là **raccoon** (vùng tích cực) và **no_raccoon** (vùng tiêu cực).

*Hình 4.2: Minh họa tập dữ liệu sau khi xử lý*

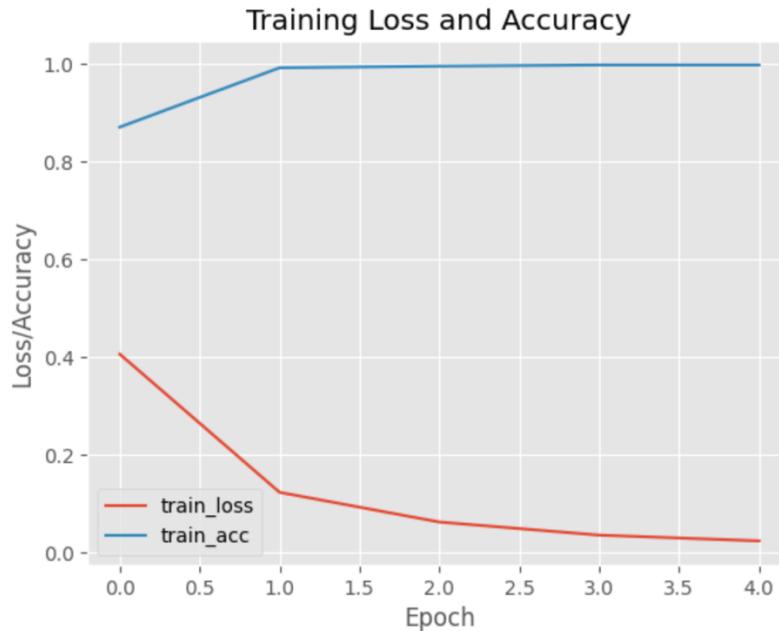
Sau khi xử lý ta có hai folder ảnh và có số lượng của hai tập:

*Hình 4.3: trực quan hóa số lượng hai lớp*

4.2.2. Train model với MobileNetV2 của Tensorflow

Sau khi tìm được hai vùng đề suất (tích cực và tiêu cực) chúng ta sẽ lưu vào hệ thống hai danh sách: danh sách data chứa các hình ảnh và danh sách labels chứa nhãn của hình đó được trích xuất từ tên của folder (raccoon và no_raccoon).

Xây dựng mạng CNN với mô hình MobileNetV2 được huấn luyện trước trên tập dữ liệu của ImageNet đã loại bỏ phần input và được thay thế bằng lớp định nghĩa mới do chúng tôi thiết lập.



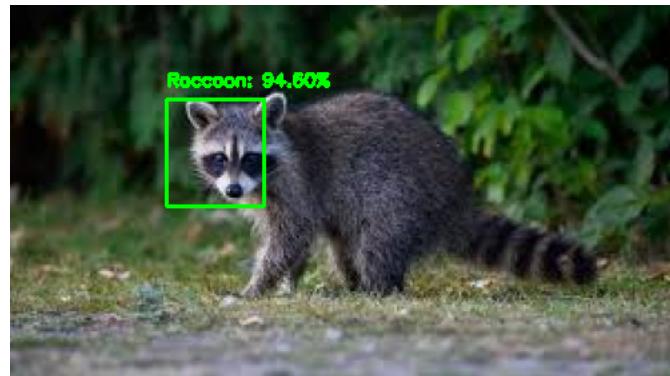
Hình 4.4: trực quan hóa kết quả huấn luyện

Nhận xét: Có thể thấy biểu đồ thể hiện hàm mất mát đã di chuyển về mức rất nhỏ và độ chính xác là 1. Điều này xảy ra trường hợp overfitting sẽ làm cho mô hình tuy có độ chính xác rất cao nhưng đối với một số ảnh thực tế mô hình sẽ không dự đoán chính xác hoàn. Lý do là dữ liệu được train không đủ lớn mặc dù đã sử dụng phương pháp tăng cường ảnh

4.2.3. Phát hiện đối tượng raccoon với R-CNN

Sau khi có được mô hình chúng tôi thử nghiệm với bức ảnh được lấy ngẫu nhiên trên internet và thu về kết quả sau qua các bước xử lý.

Kết quả lần 1:



Hình 4.5: Thực nghiệm kết quả lần 1

Kết quả lần 2:



Hình 4.6: Thực nghiệm kết quả lần 2

Kết quả lần 3:



Hình 4.7: Thực nghiệm kết quả lần 3

Kết quả lần 4:

Hình 4.8: Thực nghiệm kết quả lần 4

Nhận xét:

Dựa trên bốn lần thực nghiệm kết quả thu được các hình ảnh ta có thể nhận xét :

- **Hình 1, 3, 4:** Mô hình nhận diện chính xác đối tượng gấu mèo, cho thấy khả năng nhận diện tốt trong trường hợp đơn giản với chỉ một đối tượng rõ ràng.
- **Hình 2:** Hình này có hai con vật khác nhau, và mô hình đã xác định được gấu mèo chính xác nhưng vị trí của bounding box chưa hoàn toàn khớp với đối tượng. Điều này có thể cho thấy mô hình đã học cách phân biệt giữa raccoon và các loài vật khác, nhưng có thể gặp khó khăn trong việc tối ưu vị trí hoặc kích thước của bounding box.

CHƯƠNG 5. KẾT LUẬN

5.1. Kết quả đạt được

Mô hình nhận diện đối tượng raccoon đã đạt được kết quả khá chính xác. Trong các thử nghiệm với ảnh có raccoon, mô hình đã:

- Xác định đúng raccoon hình ảnh thử nghiệm, với một số trường hợp bounding box chưa hoàn toàn khớp với đối tượng.
- Có khả năng phân biệt raccoon khỏi các đối tượng khác trong ảnh, ngay cả khi trong ảnh xuất hiện nhiều vật thể khác nhau.

5.2. Ưu và nhược điểm

5.2.1. Ưu điểm

Khả năng nhận diện chính xác: Mô hình đã xác định đúng raccoon trong phần lớn các trường hợp thử nghiệm, cho thấy khả năng học tập và phân loại tốt trong nhiệm vụ cụ thể này.

Phân biệt đối tượng khác nhau: Mô hình có khả năng xác định raccoon trong ảnh có chứa các vật thể khác, chứng tỏ tính phân loại hiệu quả.

Xử lý tốt trong điều kiện tiêu chuẩn: Với các hình ảnh có raccoon rõ ràng và ít nhiễu, mô hình hoạt động tốt, cung cấp kết quả đáng tin cậy.

5.2.2. Nhược điểm

Vị trí bounding box chưa tối ưu: Trong một số trường hợp, như hình ảnh có hai vật thể, bounding box không hoàn toàn khớp với raccoon, cho thấy mô hình còn thiếu độ chính xác khi xác định ranh giới của đối tượng.

Thiếu độ chính xác với các ảnh phức tạp: Khi có nhiều vật thể, hoặc khi raccoon không rõ ràng, mô hình gặp khó khăn trong việc tối ưu vị trí và kích thước của bounding box.

Thời gian xử lý lâu: Khi trích xuất các vùng ROI, với 200 ảnh mất hơn 13 phút, cho thấy việc chuẩn bị dữ liệu cần nhiều thời gian. Dự đoán với 300 vùng ROI của một ảnh mất khoảng 13 giây, làm giảm hiệu quả khi xử lý số lượng lớn ảnh.

5.2.3. Hướng mở rộng trong tương lai

Tối ưu hóa bounding box: Cải thiện thuật toán non-max suppression hoặc thử các phương pháp khác như soft-non-max suppression để đạt độ chính xác cao hơn khi xác định vị trí và kích thước của bounding box.

Tăng dữ liệu: Tăng cường dữ liệu để đảm bảo cân bằng dữ liệu giữa các lớp và tăng khả năng dự đoán chính xác của mô hình.

Tích hợp và tối ưu hóa với các kiến trúc tiên tiến: Xem xét sử dụng hoặc chuyển đổi mô hình sang các kiến trúc mới như EfficientDet hoặc YOLO cho các bài toán yêu cầu độ chính xác, hiệu suất cao hơn và phù hợp với các bài toán thời gian thực.

TÀI LIỆU THAM KHẢO

Thuật toán R-CNN: <https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html>

Thuật toán R-CNN trong phát hiện đối tượng: <https://pyimagesearch.com/2020/07/13/r-cnn-object-detection-with-keras-tensorflow-and-deep-learning/>

Mô hình MobileNetV2: https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2

Silde bài giảng của thầy PGS.TS Nguyễn Thành Bình

Tập dữ liệu raccoon: <https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9>

PHỤ LỤC CODE DEMO

1) Cấu trúc dự án

📁 data_raccoon	me	Oct 8, 2024 me	—
📁 images	me	Oct 11, 2024 me	—
📁 Model	me	Nov 8, 2024 me	—
🖼️ cat.jpg	me	Oct 11, 2024 me	34 KB
🖼️ o.jpg	me	Oct 12, 2024 me	9 KB
🐍 R-CNN_Test.ipynb	me	5:29 PM me	2.1 MB
🐍 R-CNN.ipynb	me	4:49 PM me	552 KB
🖼️ raccoon_o.jpg	me	Oct 12, 2024 me	5 KB
🖼️ rc_test.jpg	me	Nov 12, 2024 me	8 KB
🖼️ ro.jpg	me	Oct 12, 2024 me	13 KB

File R-CNN.ipynb: Dùng để xử lý dữ liệu và build Model

File R-CNN_Test.ipynb: Dùng để thực nghiệm kết quả

File .jpg cat, o, raccoon_o, rc_test, ro: các bức ảnh dùng để thực nghiệm

Folder data_raccoon: Chứa file .csv có các bound box của ảnh

Folder Image: Chứa 2 folder là:

- raccoon: ảnh gốc được tải về
- img_model: Chứa 2 folder:
 - raccoon: Chứa các ảnh đè xuất có chứa raccoon được đè xuất từ tập gốc
 - no_raccoon: Chứa các ảnh đè xuất không chứa raccoon được đè xuất từ tập gốc

Folder Model: Chứa file .keras là model được lưu lại và file .pickle là nhãn được lưu lại

2) Import các thư viện cần thiết

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import os
5 import argparse
6 import pickle
7 from google.colab.patches import cv2_imshow
8 from imutils import paths
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D, Dropout, Flatten, Dense, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

Các thư viện cơ bản đã được chú thích trong phần 4.1.1 và ngoài ra sử dụng một số thư viện khác:

- *ImageDataGenerator* : Với mục đích tăng cường dữ liệu.
- *MobileNetV2* : Kiến trúc MobileNet CNN là phổ biến, vì vậy nó được tích hợp vào TensorFlow/Keras. Với mục đích tinh chỉnh, chúng tôi sẽ tải mạng bằng các trọng lượng ImageNet được đào tạo trước, cắt đầu mạng và thay thế nó, đồng thời điều chỉnh/đào tạo cho đến khi mạng của chúng tôi hoạt động tốt.
- *tensorflow.keras.layers* : Một lựa chọn các loại lớp CNN được sử dụng để xây dựng/thay thế phần đầu của MobileNet V2.
- *Adam* : Một giải pháp thay thế tối ưu hóa cho Stochastic Gradient Descent (SGD).
- *LabelBinarizer* và *to_categorical* : Được sử dụng cùng với nhau để thực hiện mã hóa một lần của các nhãn lớp.
- *train_test_split* : Thuận tiện giúp phân đoạn tập dữ liệu thành các bộ train và test
- *classification_report* : Thống kê về kết quả đánh giá mô hình

3) Khởi tạo các tham số

```
ORIG_BASE_PATH = '/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/images/raccoon'  
BASE_PATH = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/images/img_model" # folder chứa 2 lớp có raccoon và không raccoon  
  
POSITIVE_PATH = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/images/img_model/raccoon" # Hình có raccoon đã đề xuất  
NEGATIVE_PATH = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/images/img_model/no_raccoon" # Hình không raccoon đã đề xuất  
ORIG_ANNOTS = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/data_raccoon/raccoon_labels.csv" # Các tọa độ bounding box của từ
```

ORIG_BASE_PATH: đường dẫn đến folder chứa các ảnh gốc có chứa gấu mèo

BASE_PATH: đường dẫn đến folder chứa 2 folder là raccoon và no_raccoon

POSITIVE_PATH: đường dẫn đến folder chứa ảnh raccoon đã được đề xuất

NEGATIVE_PATH: đường dẫn đến folder chứa ảnh no_raccoon đã được đề xuất

ORIG_ANNOTS: đường dẫn đến file .csv chứa tọa độ hộp của các ảnh trong

ORIG_BASE_PATH

4) Xử lý ảnh

```
# grab all image paths in the input images directory  
imagePaths = list(paths.list_images(ORIG_BASE_PATH))  
  
# Lưu lại danh sách ảnh  
list_img = []  
for i in imagePaths:  
    img = cv2.imread(i)  
    list_img.append(img)  
  
  
all_gtBoxes = []  
  
for (i, imagePath) in enumerate(imagePaths):  
    filename = os.path.basename(imagePath)  
    # Lọc các bounding boxes tương ứng với ảnh hiện tại  
    selected_boxes = data_label_raccoon[data_label_raccoon.filename == filename]  
    # Danh sách để lưu bounding boxes của ảnh này  
    gtBoxes = []  
  
    # Lặp qua từng bounding box và thêm vào gtBoxes  
    for index, row in selected_boxes.iterrows():  
        xMin = int(row['xmin'])  
        yMin = int(row['ymin'])  
        xMax = int(row['xmax'])  
        yMax = int(row['ymax'])  
        # Thêm bounding box vào danh sách gtBoxes  
        gtBoxes.append((xMin, yMin, xMax, yMax))  
  
    # Thêm danh sách gtBoxes của ảnh hiện tại vào all_gtBoxes  
    all_gtBoxes.append(gtBoxes)  
  
print(len(all_gtBoxes))
```

Hàm tính độ đo IoU để tìm các vùng Rois so với các hộp thật từ file .csv

```

def compute_iou(boxA, boxB):
    # Tính tọa độ giao nhau giữa hai hộp giới hạn
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    # Tính diện tích vùng giao nhau
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    # Tính diện tích của từng hộp giới hạn
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)

    # Tính IoU = (diện tích giao) / (diện tích hợp)
    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou

```

5) Thực hiện tìm các các vùng tích cực và tiêu cực của tất cả ảnh

```

MAX_PROPOSALS = 2000 # giới hạn vùng đề xuất
MAX_POSITIVE = 30    # giới hạn vùng tích cực được lưu
MAX_NEGATIVE = 30    # giới hạn vùng tiêu cực được lưu
INPUT_DIMS = (224, 224) # kích thước ảnh được truyền vào mạng MobileNet

```

Trích xuất các vùng Rois từ bức ảnh bằng cách tìm kiếm có chọn lọc và lấy tối đa 2000 vùng. Sau đó tính độ IoU và so sánh với hộp thật mỗi vùng Rois được trích xuất từ ảnh lưu tối đa 30 hình, mỗi hình lưu lại được resize về kích thước (224, 244):

- IoU > 0.7 : được lưu vào folder raccoon
- IoU <= 0.7 and IoU >= 0.05 : được lưu vào folder no_raccoon

```

totalPositive = 0
totalNegative = 0
count_img = 0

# Duyệt qua từng ảnh và bounding boxes tương ứng
for i, (img, gtBoxes) in enumerate(zip(list_img, all_gtBoxes)):
    # Khởi tạo bộ đếm cho ROIs tích cực và tiêu cực cho ảnh hiện tại
    positiveROIs = 0
    negativeROIs = 0
    count_img += 1
    print("Processing image", count_img, "/200")
    # Khởi tạo công cụ Selective Search
    ss = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
    ss.setBaseImage(img)
    ss.switchToSelectiveSearchFast()

    # Lấy các vùng đề xuất (ROI)
    proposedRects = ss.process()

```

```

for proposedRect in proposedRects[:MAX_PROPOSALS]:
    (propStartX, propStartY, propEndX, propEndY) = proposedRect

    roi = None
    outputPath = None
    isPositive = False

    # Duyệt qua tất cả các hộp giới hạn chuẩn (ground-truth bounding boxes) cho ảnh hiện tại
    for gtBox in gtBoxes:
        iou = compute_iou(gtBox, proposedRect)
        (gtStartX, gtStartY, gtEndX, gtEndY) = gtBox
        # Lưu lại những vùng có độ IoU > 0.7 và dưới 30 vùng
        if iou > 0.7 and positiveROIs < MAX_POSITIVE:
            roi = img[propStartY:propEndY, propStartX:propEndX]
            filename = "{}.jpg".format(totalPositive)
            outputPath = os.path.sep.join([POSITIVE_PATH, filename])

            positiveROIs += 1
            totalPositive += 1
            isPositive = True
            break

        if not isPositive:
            isNegative = True
            for gtBox in gtBoxes:
                iou = compute_iou(gtBox, proposedRect)
                fullOverlap = (propStartX >= gtStartX and propStartY >= gtStartY and
                               propEndX <= gtEndX and propEndY <= gtEndY)

                if iou >= 0.05 or fullOverlap:
                    isNegative = False
                    break

if isNegative and negativeROIs < MAX_NEGATIVE:
    roi = img[propStartY:propEndY, propStartX:propEndX]
    filename = "{}.jpg".format(totalNegative)
    outputPath = os.path.sep.join([NEGATIVE_PATH, filename])

    negativeROIs += 1
    totalNegative += 1

    if roi is not None and outputPath is not None:
        if roi.size > 0:
            roi = cv2.resize(roi, INPUT_DIMS, interpolation=cv2.INTER_CUBIC)
            cv2.imwrite(outputPath, roi)

```

6) Xây dựng mô hình

Khởi tạo các tham số, đọc và lưu ảnh cùng labels được trích xuất từ tên folder.

```

INIT_LR = 1e-4
EPOCHS = 5
BS = 32

print("[INFO] loading images...")
imagePaths = list(paths.list_images(BASE_PATH))

data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]
    # load the input image (224x224) and preprocess it
    image = load_img(imagePath, target_size=INPUT_DIMS)
    image = img_to_array(image)
    image = preprocess_input(image)
    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)

[INFO] loading images

```

Xử lý ảnh và nhãn theo định dạng của MobileNet:

- raccoon: [0. 1.]
- no_raccoon: [1. 0.]

Sử dụng ImageDataGenerator để tăng cường dữ liệu ảnh để huấn luyện.

```

data = np.array(data, dtype="float32")
labels = np.array(labels)
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2, stratify=labels, random_state=42)

aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

```

Khởi tạo mạng MobileNet được huấn luyện trước trên tập dữ liệu của ImageNet đã loại bỏ phần input và được thay thế bằng input mới (224, 224, 3).

Xây dựng các tầng mạng bao gồm:

- **AveragePooling2D(pool_size=(7, 7))**: Lớp này giảm chiều kích thước của đặc trưng từ MobileNetV2 bằng cách thực hiện phép gộp trung bình, giúp giảm số lượng tham số và tăng tốc độ huấn luyện.
- **Flatten(name="flatten")**: Biến đổi đặc trưng 2D từ lớp pooling thành dạng 1D để có thể đưa vào các lớp fully-connected.
- **Dense(128, activation="relu")**: Đây là một lớp fully-connected với 128 nơ-ron, sử dụng hàm kích hoạt ReLU. Lớp này học các đặc trưng phức tạp hơn và chuẩn bị cho đầu ra cuối.
- **Dropout(0.5)**: Dropout ngẫu nhiên tắt 50% số nơ-ron trong quá trình huấn luyện, giúp giảm overfitting.
- **Dense(2, activation="sigmoid")**: Đây là lớp đầu ra với 2 nơ-ron đại diện cho hai lớp "raccoon" và "no_raccoon". Hàm kích hoạt sigmoid được sử dụng để đưa ra xác suất cho mỗi lớp.

```
# Nạp MobileNetV2 loại bỏ các lớp Top (Head)
baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor= Input(shape=(224, 224, 3)) )

# Định nghĩa lớp Top mới để thay thế lớp Top đã loại bỏ của MobileNetV2
NewTop = baseModel.output
NewTop = AveragePooling2D(pool_size=(7, 7))(NewTop)
NewTop = Flatten(name="flatten")(NewTop)
NewTop = Dense(128, activation="relu")(NewTop)
NewTop = Dropout(0.5)(NewTop)
NewTop = Dense(2, activation="sigmoid")(NewTop) # Cập nhật kích thước đầu ra theo số lớp

# Tạo model bằng cách thay thế Top từ Top mới đã định nghĩa ở trên
model = Model(inputs=baseModel.input, outputs=NewTop)

# Đóng băng lớp cơ sở (base)
for layer in baseModel.layers:
    layer.trainable = False

# Biên dịch model
print("[INFO] Biên dịch model...")
opt = Adam(learning_rate=INIT_LR)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train phần Top mới của network
print("[INFO] Đang huấn luyện phần Top (head) của mạng...")

H1 = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    epochs=EPOCHS
)
```

7) Lưu model

```

# define the path to the output model and label binarizer
MODEL_PATH = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/Model/raccoon_detector.keras"
ENCODER_PATH = "/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/Model/label_encoder.pickle"

print("[INFO] saving mask detector model...")
model.save(MODEL_PATH)
# serialize the label encoder to disk
print("[INFO] saving label encoder...")
f = open(ENCODER_PATH, "wb")
f.write(pickle.dumps(lb))
f.close()

```

8) Load model để thực hiện thực nghiệm

Khởi tạo các tham số đầu vào và load ảnh thực nghiệm

```

# Đường dẫn đến các model
MODEL_PATH_NEW = '/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/Model/raccoon_detector.keras'
ENCODER_PATH_NEW = '/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/Model/label_encoder.pickle'

model = load_model(MODEL_PATH_NEW)
lb = pickle.loads(open(ENCODER_PATH_NEW, "rb").read())

MAX_PROPOSALS = 300
MAX_POSITIVE = 30
MAX_NEGATIVE = 10
MIN_PROBA = 0.8
INPUT_DIMS = (224, 224)

path_img = '/content/drive/MyDrive/Colab Notebooks/Computer Vision/R-CNN_with_Raccoon/rc_test.jpg'
image = cv2.imread(path_img)
image = imutils.resize(image, width=500)

cv2.imshow(image)

```

9) Thuật toán NMS (non-maximum suppression)

```
def non_max_suppression(boxes, probs=None, overlapThresh=0.5):
    # if there are no boxes, return an empty list
    if len(boxes) == 0:
        return []

    # if the bounding boxes are integers, convert them to floats -- this
    # is important since we'll be doing a bunch of divisions
    if boxes.dtype.kind == "i":
        boxes = boxes.astype("float")

    # initialize the list of picked indexes
    pick = []

    # grab the coordinates of the bounding boxes
    x1 = boxes[:, 0]
    y1 = boxes[:, 1]
    x2 = boxes[:, 2]
    y2 = boxes[:, 3]

    # compute the area of the bounding boxes and grab the indexes to sort
    # (in the case that no probabilities are provided, simply sort on the
    # bottom-left y-coordinate)
    area = (x2 - x1 + 1) * (y2 - y1 + 1)
    idxs = y2

    # if probabilities are provided, sort on them instead
    if probs is not None:
        idxs = probs

    # sort the indexes
    idxs = np.argsort(idxs)
```

```

# keep looping while some indexes still remain in the indexes list
while len(idxs) > 0:
    # grab the last index in the indexes list and add the index value
    # to the list of picked indexes
    last = len(idxs) - 1
    i = idxs[last]
    pick.append(i)

    # find the largest (x, y) coordinates for the start of the bounding
    # box and the smallest (x, y) coordinates for the end of the bounding
    # box
    xx1 = np.maximum(x1[i], x1[idxs[:last]])
    yy1 = np.maximum(y1[i], y1[idxs[:last]])
    xx2 = np.minimum(x2[i], x2[idxs[:last]])
    yy2 = np.minimum(y2[i], y2[idxs[:last]])

    # compute the width and height of the bounding box
    w = np.maximum(0, xx2 - xx1 + 1)
    h = np.maximum(0, yy2 - yy1 + 1)

    # compute the ratio of overlap
    overlap = (w * h) / area[idxs[:last]]

    # delete all indexes from the index list that have overlap greater
    # than the provided overlap threshold
    idxs = np.delete(idxs, np.concatenate(([last],
                                           np.where(overlap > overlapThresh)[0])))

# return only the bounding boxes that were picked
return boxes[pick].astype("int")

```

10) Thực nghiệm kết quả

Tìm kiếm có chọn lọc bằng selective search của cv2

```

print("[INFO] running selective search...")
ss = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
ss.setBaseImage(image)
ss.switchToSelectiveSearchFast()
rects = ss.process()

print('Total Number of Region Proposals: {}'.format(len(rects)))
img_selective = image.copy()

for (x, y, w, h) in rects:
    color = [random.randint(0, 255) for j in range(0,3)]
    cv2.rectangle(img_selective, (x, y), (x + w, y + h), color, 1)

cv2_imshow(img_selective)

```

Lấy các vùng đề xuất trong retcs (tối đa là MAX_PROPOSALS được khởi tạo trước) được cắt và lưu vào proposals. Sau đó mã hóa về dạng mảng Numpy với kiểu float32 để giảm kích thước

```
proposals = []
boxes = []
# loop over the region proposal bounding box coordinates generated by
# running selective search
for (x, y, w, h) in rects[:MAX_PROPOSALS]:
    # extract the region from the input image, convert it from BGR to
    # RGB channel ordering, and then resize it to the required input
    # dimensions of our trained CNN
    roi = image[y:y + h, x:x + w]
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)
    roi = cv2.resize(roi, INPUT_DIMS, interpolation=cv2.INTER_CUBIC)
    # further preprocess the ROI
    roi = img_to_array(roi)
    roi = preprocess_input(roi)
    # update our proposals and bounding boxes lists
    proposals.append(roi)
    boxes.append((x, y, x + w, y + h))

# clone the original image so that we can draw on it
proposals = np.array(proposals, dtype="float32")
boxes = np.array(boxes, dtype="int32")
print("[INFO] proposal shape: {}".format(proposals.shape))

clone = image.copy()
print("[INFO] shape của Vùng đề xuất sau khi resize: {}".format(proposals.shape))

img_proposal = image.copy()

for (x, y, x2, y2) in boxes:
    color = [random.randint(0, 255) for j in range(0,3)]
    cv2.rectangle(img_proposal, (x, y), (x2, y2), color, 1)

plt.imshow(img_proposal[:, :, ::-1])
print("[INFO] {} original bounding boxes".format(len(boxes)))
```

Nạp mô hình để đưa ra dự đoán để trả về các vùng đề xuất có giá trị xác suất cao

Sau đó, tìm các vùng đề xuất có giá trị lớn nhất trên trực thứ nhất (hàng) của proba, tức là dự đoán lớp có xác suất cao nhất cho mỗi ROI. Tiếp theo, xáo trộn thứ tự các đề xuất ROI. Sau đó, chỉ lấy chỉ số của các boxes có xác suất cao hơn MIN_PROB và thuộc lớp ‘raccoon’.

```

print("[INFO] Nạp mô hình tinh chỉnh và file mã hóa nhãn, và gán nhãn cho Vùng để xuất")
proba = model.predict(proposals)

[INFO] Nạp mô hình tinh chỉnh và file mã hóa nhãn, và gán nhãn cho Vùng để xuất
10/10 ━━━━━━━━ 14s 1s/step

# Tìm các dự đoán dương tính cho lớp "raccoon" với xác suất >= MIN_PROBA
labels = lb.classes_[np.argmax(proba, axis=1)]
idxs = np.where((labels == "raccoon") & (proba[:, 1] >= MIN_PROBA))[0]

# Xáo trộn và giới hạn số lượng proposals
random.shuffle(proposals)
proposals = proposals[:len(boxes)]

# Lọc lại boxes và proba chỉ với các dự đoán "raccoon" đạt ngưỡng xác suất
boxes = boxes[idxs]
proba = proba[idxs][:, 1]

```

Thực hiện hàm nms để tìm ra vùng tốt nhất để vẽ hộp giới hạn cho đối tượng và hiển thị ra màn hình.

```

boxIdxs = non_max_suppression(boxes, proba, overlapThresh = 0.9)
print("[INFO] after non-maximum suppression: {}".format(boxIdxs.shape))

img_output = image.copy()

# Iterate through each element of boxIdxs
for i in boxIdxs.flatten(): # or boxIdxs.ravel()
    if 0 <= i < len(boxes): # Check if index is within bounds
        # draw the bounding box, label, and probability on the image
        (startX, startY, endX, endY) = boxes[i]
        cv2.rectangle(img_output, (startX, startY), (endX, endY), (0, 255, 0), 2)
        y = startY - 10 if startY - 10 > 10 else startY + 10
        text = "Raccoon: {:.2f}%".format(proba[i] * 100)
        cv2.putText(img_output, text, (startX, y), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)

print(text)
# show the output image *after* running NMS
cv2.imshow(img_output)

```