

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP LỚN

HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

**Đề tài số 13: xây dựng hệ thống nhận dạng và phân loại động vật ăn
thịt và ăn cỏ trong ảnh**

Giảng viên: Lương Thị Hồng Lan

TT	Mã sinh viên	Sinh viên thực hiện	Lớp hành chính
1	20211617	Phạm Ngọc Anh	DCCNTT12.10.4
2	20211177	Hoàng Văn Thống	DCCNTT12.10.4
3	20213652	Phạm Văn Bắc	DCCNTT12.10.4
4	20210945	Nguyễn Tuấn Anh	DCCNTT12.10.4
5	20211054	Nguyễn Thị Loan	DCCNTT12.10.4

Bắc Ninh, năm 2024

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

**Đề tài số 13: xây dựng hệ thống nhận dạng và phân loại động vật ăn
thịt và ăn cỏ trong ảnh**

Giảng viên: Lương Thị Hồng Lan

TT	Mã sinh viên	Sinh viên thực hiện	Lớp hành chính
1	20211617	Phạm Ngọc Anh	DCCNTT12.10.4
2	20211177	Hoàng Văn Thống	DCCNTT12.10.4
3	20213652	Phạm Văn Bắc	DCCNTT12.10.4
4	20210945	Nguyễn Tuấn Anh	DCCNTT12.10.4
5	20211054	Nguyễn Thị Loan	DCCNTT12.10.4

Bắc Ninh, năm 2024

PHIẾU CHẤM THI BÀI TẬP LỚN KẾT THÚC HỌC PHẦN

Mã đề thi: **13**

Tên học phần: Xử lý ảnh và thị giác máy tính

Lớp Tín chỉ: XATGMT.03.K12.04.LH.C04.1_LT.1_TH

Cán bộ chấm thi 1

(Ký và ghi rõ họ tên)

Cán bộ chấm thi 2

(Ký và ghi rõ họ tên)

TT	TIÊU CHÍ	THANG ĐIỂM	Phạm Ngọc Anh	Hoàng Văn Thống	Phạm Văn Bắc	Nguyễn Tuấn Anh	Nguyễn Thị Loan
			20211617	20211177	20213652	20210945	20211054
1	Nội dung báo cáo trên Word đầy đủ	3,5					
1.1	Có bố cục rõ ràng (mục lục, phần mở đầu, nội dung chính, kết luận).	0,5					
1.2	Nội dung phân tích rõ ràng, logic.	0,5					
1.3	Có dẫn chứng, số liệu minh họa đầy đủ.	0,5					
1.4	Ngôn ngữ và trình bày chuẩn, không lỗi chính tả.	0,5					
1.5	Có trích dẫn tài liệu tham khảo đúng quy cách.	0,5					

TT	TIÊU CHÍ	THANG ĐIỂM	Phạm Ngọc Anh	Hoàng Văn Thống	Phạm Văn Bắc	Nguyễn Tuấn Anh	Nguyễn Thị Loan
			20211617	20211177	20213652	20210945	20211054
1.6	Được trình bày chuyên nghiệp (canh lề, font chữ, khoảng cách dòng hợp lý).	0,5					
1.7	Tài liệu đầy đủ, bám sát yêu cầu của đề bài.	0,5					
2	Nội dung thuyết trình đầy đủ	1.0					
2.1	Trình bày tự tin, phát âm rõ ràng, mạch lạc.	0,5					
2.2	Nội dung thuyết trình đúng trọng tâm, không lan man.	0,5					
3	Slides báo cáo đầy đủ nội dung + Hỏi đáp	3.0					
3.1	Slides có bố cục rõ ràng (mở đầu, nội dung, kết luận).	0,5					
3.2	Thiết kế slides đẹp, chuyên nghiệp (màu sắc, hình ảnh minh họa).	0,5					
3.3	Nội dung trên slides ngắn gọn, dễ hiểu, súc tích.	0,5					
3.4	Nội dung slides phù hợp với nội dung báo cáo.	0,5					
3.5	Trả lời câu hỏi đầy đủ, chính xác.	0,5					
3.6	Trả lời câu hỏi tự tin, thuyết phục.	0,5					
4	Code đầy đủ	2.5					
1.1	Code được trình bày rõ ràng, có chú thích đầy đủ.	0,5					
1.2	Code chạy đúng, không lỗi.	0,5					

TT	TIÊU CHÍ	THANG ĐIỂM	Phạm Ngọc Anh	Hoàng Văn Thống	Phạm Văn Bắc	Nguyễn Tuấn Anh	Nguyễn Thị Loan
			20211617	20211177	20213652	20210945	20211054
1.3	Code tối ưu, không dư thừa.	0,5					
1.4	Đáp ứng đầy đủ các yêu cầu chức năng theo đề bài.	0,5					
1.5	Có tính sáng tạo hoặc cải thiện so với yêu cầu.	0,5					
TỔNG ĐIỂM BẰNG SỐ:		10					
TỔNG ĐIỂM BẰNG CHỮ:		Mười tròn					

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ ngày càng phát triển, xử lý ảnh và thị giác máy tính đã trở thành một trong những lĩnh vực trọng tâm, mang lại nhiều ứng dụng đột phá trong đời sống và khoa học. Đặc biệt, việc áp dụng các kỹ thuật hiện đại để nhận dạng và phân loại động vật không chỉ giúp hỗ trợ nghiên cứu khoa học mà còn góp phần quan trọng trong việc bảo vệ và giám sát môi trường sống tự nhiên.

Đề tài “Xây dựng hệ thống nhận dạng và phân loại động vật ăn thịt và ăn cỏ trong ảnh” là một nỗ lực nhằm tận dụng những tiến bộ trong lĩnh vực học máy (Machine Learning) và học sâu (Deep Learning) để giải quyết một bài toán thực tiễn, đó là phân loại động vật theo tập tính ăn uống dựa trên dữ liệu ảnh. Hệ thống không chỉ giúp nâng cao hiệu quả nghiên cứu sinh học mà còn mở rộng khả năng ứng dụng trong các lĩnh vực như bảo tồn thiên nhiên, giáo dục, và công nghệ giám sát tự động.

Dưới sự hướng dẫn tận tình của cô **Lương Thị Hồng Lan**, nhóm thực hiện đề tài đã áp dụng các phương pháp từ xử lý ảnh, trích xuất đặc trưng đến thiết kế và huấn luyện mô hình phân loại. Bằng việc ứng dụng các công nghệ hiện đại và triển khai thực nghiệm, hệ thống hứa hẹn đạt được độ chính xác và hiệu quả cao, đáp ứng các yêu cầu đặt ra.

Với lòng nhiệt huyết và tinh thần học hỏi, chúng tôi hy vọng đề tài sẽ không chỉ là bài học ý nghĩa trong khuôn khổ học phần Xử lý ảnh và thị giác máy tính, mà còn góp phần tạo tiền đề cho những nghiên cứu và ứng dụng sâu hơn trong tương lai.

MỤC LỤC

DANH MỤC HÌNH ẢNH, BẢNG BIỂU	3
CHƯƠNG I: KIẾN THỨC CƠ SỞ	4
1.1 Nhận dạng	4
1.1.1 Các khía cạnh của nhận dạng	4
1.1.2. Ứng dụng của nhận dạng:	5
1.2 Các phương pháp sử dụng trong nhận dạng	7
1.2.1. Sử dụng đặc trưng ảnh	7
1.2.2. Các mô hình nơron	7
1.3 Ngôn ngữ lập trình và thư viện	8
1.3.1. Giới thiệu ngôn ngữ lập trình	8
1.3.2. Các thư viện	10
1.4 Các kỹ thuật	22
1.4.1. HOG (Histogram of Oriented Gradients)	22
1.4.2. SVM (Support Vector Machine)	25
1.4.3. YOLO (You Only Look Once)	29
CHƯƠNG II: XÂY DỰNG HỆ THỐNG	34
2.1. Bài toán nhận dạng và phân loại động vật ăn thịt và ăn cỏ trong ảnh	34
2.1.1. Giới thiệu bài toán	34
2.1.2. Mục tiêu của bài toán	34
2.1.3. Dữ liệu ảnh đầu vào (input)	35
2.1.4. Dữ liệu ảnh đầu ra (output)	35
2.1.5. Ý nghĩa và đóng góp của đề tài	36
2.2. Xây dựng hệ thống	36
2.2.1. Cấu trúc project	36

2.2.2. Quy trình thực thi	37
2.2.3 Tiền xử lý ảnh	38
2.2.4 Phát hiện đối tượng với YOLOv8.....	38
2.2.5. Trích xuất đặc trưng HOG từ các vùng đối tượng	39
2.3.6. Phân loại động vật bằng SVM	39
2.3.7. Hiển thị kết quả	39
CHƯƠNG III: KẾT QUẢ THỰC NGHIỆM	40
3.1. Dữ liệu.....	40
3.2 Độ đo đánh giá	41
3.2.1. Accuracy	41
3.2.2. Precision	42
3.2.3. Recall.....	43
3.2.4. F1-Score	44
3.2.4. Thời gian xử lý.....	45
3.2. Kết quả thực nghiệm	45
3.3.1. Kết quả thực nghiệm SVM	45
3.3.2. Một số hình ảnh kết quả output.....	46
KẾT LUẬN.....	49
TÀI LIỆU THAM KHẢO.....	51

DANH MỤC HÌNH ẢNH, BẢNG BIỂU

<i>Bảng 1: Bảng mô tả cấu trúc project.....</i>	<i>36</i>
<i>Bảng 2: Dữ liệu input được huấn luyện và kiểm tra</i>	<i>40</i>
<i>Bảng 3: SVM Classification Report:</i>	<i>45</i>
<i>Hình ảnh 1: Ứng dụng về thị giác máy tính</i>	<i>5</i>
<i>Hình ảnh 2: Ứng dụng về xử lý giọng nói</i>	<i>5</i>
<i>Hình ảnh 3: Ứng dụng về sinh trắc học</i>	<i>6</i>
<i>Hình ảnh 4: Ứng dụng về nhận dạng văn bản</i>	<i>6</i>
<i>Hình ảnh 5: Mô hình minh họa cấu trúc hỗ trợ ngôn ngữ Python.....</i>	<i>9</i>
<i>Hình ảnh 6: Mô hình minh họa các thư viện hỗ trợ của ngôn ngữ Python.....</i>	<i>20</i>
<i>Hình ảnh 7: hình minh họa phân loại văn bản.....</i>	<i>28</i>
<i>Hình ảnh 8: Hình minh họa nhận dạng ảnh.....</i>	<i>29</i>
<i>Hình ảnh 9: Hình minh họa phân tích sinh học</i>	<i>29</i>
<i>Hình ảnh 10: Hình minh họa ảnh động vật input.....</i>	<i>35</i>
<i>Hình ảnh 11:Hình ảnh động vật đã được nhận dạng đúng yêu cầu output</i>	<i>35</i>
<i>Hình ảnh 12: Hình ảnh các foder nhận dữ liệu động vật.....</i>	<i>40</i>
<i>Hình ảnh 13: Kết quả output đầu ra mô hình (1).....</i>	<i>46</i>
<i>Hình ảnh 13: Kết quả output đầu ra mô hình (2).....</i>	<i>47</i>
<i>Hình ảnh 14: Kết quả output đầu ra mô hình (3).....</i>	<i>47</i>
<i>Hình ảnh 15: Kết quả output đầu ra mô hình (4).....</i>	<i>48</i>
<i>Hình ảnh 16: Kết quả output đầu ra mô hình (5).....</i>	<i>48</i>

CHƯƠNG I: KIẾN THỨC CƠ SỞ

1.1 Nhận dạng

Nhận dạng (Recognition) là quá trình xử lý dữ liệu nhằm xác định, phân loại hoặc phân biệt các đối tượng, mẫu hoặc thông tin cụ thể dựa trên các đặc điểm nhận dạng đặc trưng của chúng. Đây là một lĩnh vực quan trọng trong khoa học máy tính, trí tuệ nhân tạo, và học máy, được áp dụng trong nhiều lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, và sinh trắc học.

1.1.1 Các khía cạnh của nhận dạng

❖ Đầu vào (Input):

- Dữ liệu cần nhận dạng có thể là hình ảnh, âm thanh, văn bản, hoặc dữ liệu số.
- Ví dụ: Một hình ảnh chứa khuôn mặt người, một đoạn âm thanh giọng nói, hoặc một chuỗi văn bản.

❖ Đặc trưng (Features):

Quá trình nhận dạng tập trung vào việc trích xuất các đặc điểm (features) nổi bật của đối tượng, ví dụ:

- Đặc điểm khuôn mặt (mắt, mũi, miệng) trong nhận dạng khuôn mặt.
- Tần số, cao độ trong nhận dạng giọng nói.
- Mẫu ký tự hoặc từ ngữ trong nhận dạng chữ viết tay.

❖ So sánh và đối chiếu:

- Sau khi trích xuất đặc điểm, hệ thống sẽ so sánh chúng với dữ liệu đã được học hoặc lưu trữ trước đó (cơ sở dữ liệu mẫu).
- Ví dụ: So sánh một dấu vân tay mới với các dấu vân tay trong cơ sở dữ liệu.

❖ Phân loại và quyết định:

- Hệ thống sẽ phân loại hoặc đưa ra quyết định về đối tượng dựa trên mức độ tương đồng hoặc dự đoán xác suất.

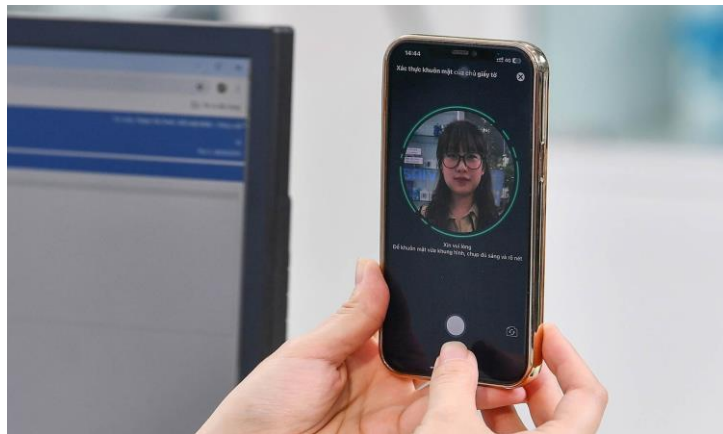
- Kết quả thường là một nhãn (label) hoặc thông tin xác định đối tượng.

1.1.2. Ứng dụng của nhận dạng:

Ứng dụng của nhận dạng được sử dụng rộng rãi trong đời sống :

❖ Thị giác máy tính (Computer Vision):

- Nhận dạng khuôn mặt, vật thể, cảnh quan trong ảnh và video.
- Ví dụ: Ứng dụng mở khóa điện thoại bằng khuôn mặt, xe tự hành nhận dạng làn đường.



Hình ảnh 1: Ứng dụng về thị giác máy tính

❖ Xử lý giọng nói (Speech Processing):

- Nhận dạng giọng nói để chuyển đổi âm thanh thành văn bản (speech-to-text).
- Ví dụ: Trợ lý ảo như Siri, Google Assistant.



Hình ảnh 2: Ứng dụng về xử lý giọng nói

❖ Sinh trắc học (Biometrics):

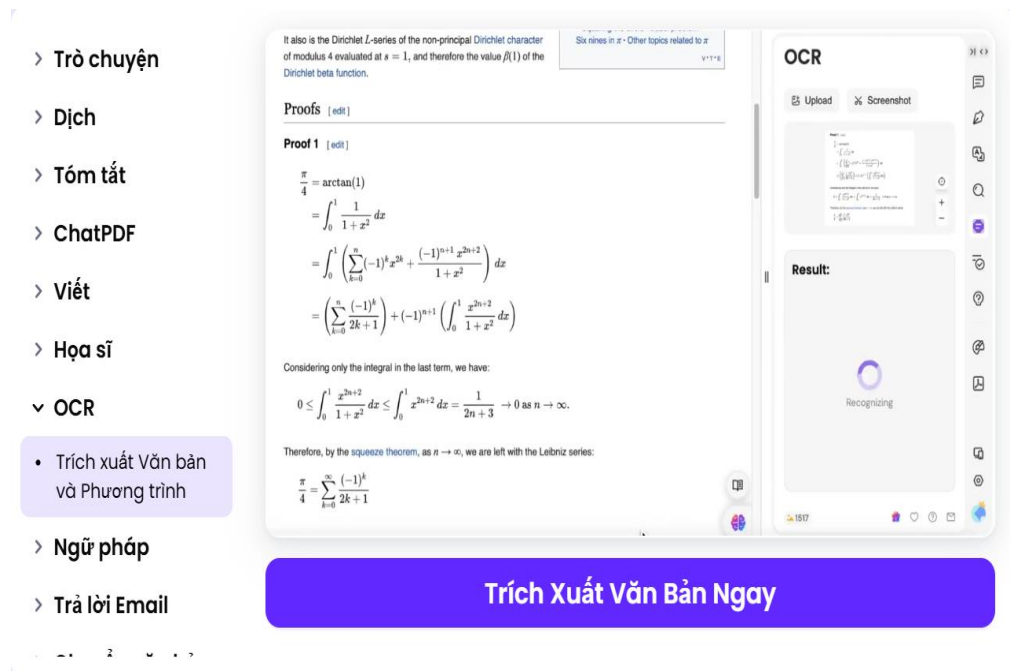
- Nhận dạng vân tay, khuôn mặt, mống mắt để xác thực danh tính.
- Ứng dụng trong an ninh, mở khóa thiết bị.



Hình ảnh 3: Ứng dụng về sinh trắc học

Nhận dạng văn bản (Text Recognition):

- Nhận dạng chữ viết tay, ký tự in, hoặc ngôn ngữ tự nhiên.
- Ví dụ: OCR để quét tài liệu, dịch tự động.



Hình ảnh 4: Ứng dụng về nhận dạng văn bản

❖ Phát hiện gian lận:

- Nhận dạng các mẫu hành vi bất thường trong giao dịch tài chính hoặc an ninh mạng.

1.2 Các phương pháp sử dụng trong nhận dạng

1.2.1. Sử dụng đặc trưng ảnh

❖ *Đặc trưng ảnh là các thông tin trích xuất từ hình ảnh, ví dụ như:*

- *Màu sắc (Color):* Phân tích màu sắc đặc trưng của lông hoặc da.
- *Hình dạng (Shape):* Nhận dạng cấu trúc cơ thể, ví dụ như đầu, chân, và đuôi.
- *Kết cấu (Texture):* Phát hiện các hoa văn trên lông hoặc vảy của động vật.

❖ *Phương pháp:*

- *HOG (Histogram of Oriented Gradients):* Dùng để phát hiện cạnh và hình dạng.
- *SIFT (Scale-Invariant Feature Transform):* Trích xuất đặc trưng không thay đổi theo kích thước và góc nhìn.
- *SURF (Speeded-Up Robust Features):* Cải tiến từ SIFT, nhanh hơn và hiệu quả hơn.

1.2.2. Các mô hình nơron

❖ Các mô hình nơron nhân tạo sử dụng học sâu (Deep Learning) để tự động học đặc trưng từ dữ liệu:

- *CNN (Convolutional Neural Networks):*
 - Mạnh mẽ trong xử lý ảnh, đặc biệt phù hợp cho nhận dạng và phân loại động vật.
 - Mô hình tiêu biểu: ResNet, VGG, Inception.
- *RNN (Recurrent Neural Networks):*
 - Xử lý dữ liệu tuần tự, ví dụ nhận dạng động vật từ video liên tục.

❖ *Phương pháp kết hợp:*

- Object Detection Models: YOLO, Faster R-CNN để phát hiện vị trí động vật trong ảnh trước khi phân loại.
- Pretrained Models: Sử dụng các mô hình đã huấn luyện trên tập dữ liệu lớn như ImageNet, COCO, rồi tinh chỉnh (fine-tuning) cho bài toán cụ thể.

1.3 Ngôn ngữ lập trình và thư viện

1.3.1. Giới thiệu ngôn ngữ lập trình

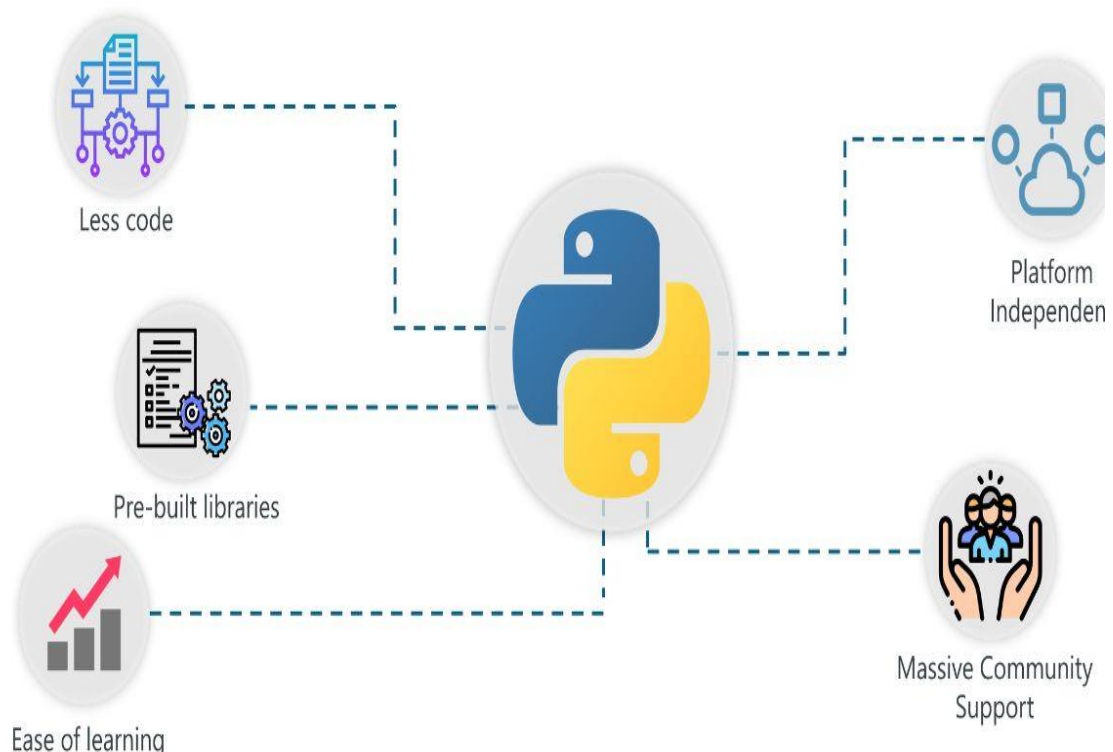
Python là một ngôn ngữ lập trình thông dịch, dễ học và sử dụng, được phát triển lần đầu tiên vào năm 1991 bởi Guido van Rossum. Python nổi bật nhờ cú pháp rõ ràng, dễ hiểu, và có cộng đồng hỗ trợ rất lớn. Đây là lý do tại sao Python rất phổ biến trong cả giáo dục lẫn phát triển phần mềm thực tế.



1.3.1.1. Đặc điểm của Python:

- ❖ Cú pháp dễ đọc: Python được thiết kế với mục tiêu làm cho mã nguồn dễ đọc và dễ viết. Cú pháp của nó rất giống với tiếng Anh, giúp người mới bắt đầu dễ dàng làm quen.
- ❖ Cấu trúc vòng lặp for trong Python dễ hiểu và không cần dấu ngoặc nhọn như các ngôn ngữ khác (như C, Java).
- ❖ Ngôn ngữ thông dịch: Python không cần biên dịch thành mã máy trước khi chạy. Mã nguồn của Python có thể chạy ngay lập tức trên môi trường thông dịch, giúp việc thử nghiệm và phát triển trở nên nhanh chóng.
- ❖ Hỗ trợ nhiều lĩnh vực ứng dụng: Python được sử dụng rộng rãi trong nhiều lĩnh vực, chẳng hạn như:
 - Phát triển web: Với các framework như Django, Flask.
 - Khoa học dữ liệu: Thư viện như Pandas, NumPy, Matplotlib giúp xử lý dữ liệu, phân tích thống kê, vẽ đồ thị.

- Máy học và AI: Thư viện như TensorFlow, PyTorch rất mạnh mẽ trong việc xây dựng các mô hình học sâu.
- Tự động hóa: Python được sử dụng để viết các script tự động hoá công việc lặp đi lặp lại.
- Phát triển phần mềm: Python có thể dùng để xây dựng ứng dụng phần mềm từ nhỏ đến lớn, từ ứng dụng desktop đến ứng dụng di động.



Hình ảnh 5: Mô hình minh họa cấu trúc hỗ trợ ngôn ngữ Python

- ❖ **Thư viện và cộng đồng lớn:** Python có một kho thư viện phong phú và cộng đồng phát triển mạnh mẽ. Bạn có thể dễ dàng tìm thấy tài liệu, hướng dẫn và mã nguồn mở cho gần như mọi nhu cầu lập trình.
- ❖ **Cộng đồng và tài nguyên học tập:** Python có một cộng đồng người dùng và lập trình viên rộng lớn, nơi bạn có thể tìm thấy rất nhiều tài liệu, diễn đàn hỗ trợ, và mã nguồn mở. Python cũng được giảng dạy trong nhiều khóa học lập trình cơ bản.

1.3.1.2. Ưu điểm của Python:

- ❖ **Dễ học:** Python là một trong những ngôn ngữ lập trình dễ học nhất, thích hợp cho người mới bắt đầu.
- ❖ **Mã nguồn mở:** Python miễn phí và mã nguồn của nó có sẵn để bạn có thể sử dụng và tùy chỉnh.
- ❖ **Khả năng mở rộng:** Python có thể dễ dàng tích hợp với các ngôn ngữ khác như C, C++ để cải thiện hiệu suất nếu cần.

1.3.1.3. Nhược điểm của Python

- ❖ **Tốc độ thực thi:** Do là ngôn ngữ thông dịch, Python có thể chậm hơn so với các ngôn ngữ biên dịch như C++ hoặc Java khi thực hiện các tác vụ đòi hỏi hiệu suất cao.
- ❖ **Quản lý bộ nhớ:** Python sử dụng Garbage Collection (GC) để quản lý bộ nhớ, điều này đôi khi có thể gây overhead nếu không được tối ưu đúng cách.

Nhìn chung, Python là một lựa chọn tuyệt vời cho người mới bắt đầu lập trình cũng như những người đã có kinh nghiệm. Với sự phát triển mạnh mẽ của cộng đồng và các thư viện hỗ trợ, Python đã trở thành một ngôn ngữ lập trình không thể thiếu trong nhiều ngành công nghiệp hiện đại.

1.3.2. Các thư viện

1.3.2.1. Xử lý ảnh:

- ❖ **Pillow (PIL - Python Imaging Library):**

Pillow là thư viện đơn giản và dễ sử dụng, được thiết kế để hỗ trợ xử lý ảnh cơ bản và trung gian.

- **Các ứng dụng chính của Pillow**

- Đọc và ghi ảnh từ nhiều định dạng khác nhau (JPEG, PNG, BMP, v.v.)
- Thực hiện các thao tác cơ bản trên ảnh như cắt, xoay, thay đổi kích thước.
- Thêm hiệu ứng đơn giản như làm mờ, chỉnh độ sáng, độ tương phản.

❖ Các chức năng tiêu biểu của Pillow

➤ *Đọc/ghi ảnh*

`Image.open()` - Đọc ảnh từ tệp.

`Image.save()` - Lưu ảnh sau khi chỉnh sửa.

➤ *Biến đổi ảnh*

`Image.resize()` - Thay đổi kích thước ảnh.

`Image.crop()` - Cắt ảnh theo vùng được chỉ định.

`Image.rotate()` - Xoay ảnh theo góc.

➤ *Xử lý màu sắc*

`ImageEnhance.Brightness()` - Điều chỉnh độ sáng.

`ImageEnhance.Contrast()` - Điều chỉnh độ tương phản.

➤ *Làm mờ*

`ImageFilter.GaussianBlur()` - Làm mờ ảnh bằng Gaussian Blur.

❖ OpenCV (Open Source Computer Vision Library):

OpenCV là một thư viện thư viện mã nguồn mở phổ biến nhất cho xử lý ảnh và video, cung cấp rất nhiều tính năng mạnh mẽ.

➤ *Các ứng dụng chính của OpenCV*

- Phát hiện và theo dõi đối tượng trong video
- Xử lý ảnh: làm mờ, tăng cường chất lượng, chỉnh sửa
- Phân tích và trích xuất thông tin từ ảnh (nhận diện khuôn mặt, biển số xe, vật thể, v.v.)

➤ *Các chức năng tiêu biểu của OpenCV*

- Phát hiện cạnh
`cv2.Canny()` - Thực hiện phát hiện cạnh bằng thuật toán Canny Edge Detection.
- *Làm mờ ảnh*
`cv2.GaussianBlur()` - Áp dụng Gaussian Blur để làm mờ ảnh và giảm nhiễu
- *Trích xuất đặc trưng*
`cv2.SIFT()` hoặc `cv2.ORB()` - Trích xuất các điểm đặc trưng quan trọng của ảnh
- *Phép biến đổi hình học*
`cv2.warpAffine()` - Xoay, dịch chuyển, hoặc thay đổi kích thước ảnh
- *Đọc/ghi ảnh*
`cv2.imread()` - Đọc ảnh từ tệp.
`cv2.imwrite()` - Lưu ảnh sau khi xử lý.

❖ Scikit-image:

Scikit-image là thư viện chuyên biệt dành cho xử lý ảnh, được xây dựng trên các thư viện như NumPy và SciPy, cung cấp các thuật toán mạnh mẽ và tối ưu cho nhiều tác vụ khác nhau.

- **Các ứng dụng chính của scikit-image**
 - Phân đoạn ảnh, phát hiện biên, và trích xuất đặc trưng nâng cao.
 - Xử lý ảnh số: lọc, tăng cường, và biến đổi hình học.
 - Phân tích hình dạng và đo lường.
- **Các chức năng tiêu biểu của scikit-image**
 - Phát hiện cạnh
`feature.canny()` - Sử dụng thuật toán Canny Edge Detection.
 - Làm mờ và lọc ảnh
`filters.gaussian()` - Làm mờ ảnh bằng Gaussian Filter.
`filters.median()` - Loại bỏ nhiễu bằng Median Filter.

- Phân đoạn ảnh
`segmentation.slic()` - Phân đoạn ảnh bằng thuật toán SLIC (Simple Linear Iterative Clustering).
- Chuyển đổi hình học
`transform.resize()` - Thay đổi kích thước ảnh.
`transform.rotate()` - Xoay ảnh theo góc cụ thể.
- Tăng cường ảnh
`exposure.adjust_gamma()` - Điều chỉnh gamma của ảnh.
`exposure.equalize_hist()` - Cân bằng histogram để tăng cường độ tương phản.

1.3.2.2. Hỗ trợ Triển khai và API

❖ Flask:

Flask là một framework Python phổ biến, nhẹ nhàng và linh hoạt, thường được sử dụng để xây dựng các ứng dụng web và API RESTful.

➤ Các tính năng chính:

- Phát triển API REST đơn giản.
- Hỗ trợ tích hợp các công cụ front-end như HTML, CSS, JavaScript.
- Hệ sinh thái mở rộng thông qua các extension như Flask-SQLAlchemy, Flask-CORS

❖ FastAPI:

Fast API là một framework hiện đại, nhanh và mạnh mẽ để xây dựng API RESTful. Nó tối ưu hóa hiệu năng và cung cấp hỗ trợ tự động cho OpenAPI (Swagger UI).

➤ Các tính năng chính

- Tích hợp sẵn Open API và JSON Schema.
- Hiệu suất cao, gần như ngang ngửa với Node.js.
- Kiểm tra kiểu dữ liệu (type checking) mạnh mẽ thông qua **Pedantic**.

❖ Streamlit:

Streamlit là một framework giúp tạo các dashboard và ứng dụng web trực quan hóa dữ liệu chỉ với vài dòng mã.

➤ Các tính năng chính:

- Triển khai giao diện nhanh chóng mà không cần kỹ năng front-end.
- Hỗ trợ tương tác dữ liệu thời gian thực.
- Tích hợp dễ dàng với mô hình học máy hoặc xử lý ảnh.

❖ Swagger/OpenAPI:

Swagger và OpenAPI là các công cụ mạnh mẽ giúp thiết kế, xây dựng và tài liệu hóa API cho các hệ thống web, bao gồm các API cung cấp mô hình AI.

➤ Các tính năng chính:

- *Tạo tài liệu API tự động*: Swagger giúp bạn tạo tài liệu API tự động từ mã nguồn, giúp các nhà phát triển dễ dàng hiểu và tích hợp API vào các hệ thống khác.
- *Kiểm thử API*: Swagger cũng hỗ trợ kiểm thử API, cho phép bạn xác minh xem API của mình có hoạt động như mong muốn hay không.

1.3.2.3 Xử lý dữ liệu và toán học cơ bản:

❖ NumPy:

NumPy là một thư viện quan trọng trong Python để xử lý các mảng số liệu (arrays) và thực hiện các phép toán cơ bản. Với NumPy, bạn có thể:

- *Xử lý mảng đa chiều*: NumPy hỗ trợ mảng n chiều (ndarray), giúp bạn lưu trữ và thao tác trên dữ liệu dạng bảng hoặc ma trận.
- *Toán học cơ bản*: NumPy cung cấp nhiều hàm toán học như cộng, trừ, nhân, chia, các phép toán ma trận, và các phép toán số học như bình phương, căn bậc hai, logarit, v.v.

- *Các phép toán tuyến tính và ma trận:* Với hàm `np.linalg`, NumPy có thể thực hiện các phép toán tuyến tính như phép nhân ma trận, nghịch đảo ma trận, xác định giá trị riêng, v.v.

❖ **Pandas:**

Pandas là thư viện mạnh mẽ cho việc xử lý và phân tích dữ liệu dạng bảng, rất thích hợp cho dữ liệu có cấu trúc hoặc có nhãn (metadata).

- *Quản lý dữ liệu bảng:* Pandas cung cấp các cấu trúc dữ liệu như DataFrame và Series, giúp dễ dàng lưu trữ và thao tác với dữ liệu dạng bảng, chẳng hạn như bảng tính hoặc cơ sở dữ liệu.
- *Phân tích dữ liệu:* Pandas có các công cụ mạnh mẽ để xử lý missing data, lọc dữ liệu, nhóm (groupby), tính toán các phép thống kê, v.v.
- *Tương tác với cơ sở dữ liệu:* Bạn có thể dễ dàng kết nối và truy vấn dữ liệu từ các cơ sở dữ liệu như SQL hoặc các file dữ liệu như CSV, Excel, JSON.

❖ **Matplotlib Seaborn:**

Đây là các thư viện chuyên dụng cho việc trực quan hóa dữ liệu.

- *Matplotlib:* Cung cấp khả năng vẽ đồ thị cơ bản như biểu đồ cột, đường, tán xạ, v.v. Bạn có thể tùy chỉnh rất nhiều tham số trong đồ thị để hiển thị theo yêu cầu.
- *Seaborn:* Là một thư viện nâng cao được xây dựng trên Matplotlib, hỗ trợ trực quan hóa dữ liệu thống kê, như biểu đồ phân phối, biểu đồ hộp, hoặc vẽ heatmap.

1.3.2.4. Học sâu và mô hình hóa:

❖ **scikit-learn:**

Scikit-learn là một thư viện Python phổ biến trong lĩnh vực Machine Learning, cung cấp các thuật toán học máy và công cụ tiền xử lý. Các tính năng chính của scikit-learn:

➤ *Thuật toán học máy cơ bản:*

- Học có giám sát: SVM (Support Vector Machine), Random Forest, Logistic Regression.
- Học không giám sát: K-Means, PCA (Principal Component Analysis).

➤ *Tiền xử lý dữ liệu:*

- Chuẩn hóa dữ liệu: Min-Max Scaler, Standard Scaler.
- Mã hóa dữ liệu: One-Hot Encoding, Label Encoding.

➤ *Công cụ đánh giá mô hình:*

- scikit-learn hỗ trợ việc chia dữ liệu thành các tập huấn luyện và kiểm tra, giúp huấn luyện và đánh giá mô hình.
- Cross-validation, GridSearchCV, và các chỉ số như accuracy, precision, recall.

❖ **TensorFlow/Keras**

:

TensorFlow là một framework mạnh mẽ dành cho Deep Learning, phát triển bởi Google Brain, thường được sử dụng trong sản xuất nhờ hiệu năng cao và khả năng triển khai đa nền tảng. **Keras** là API cấp cao của TensorFlow, giúp đơn giản hóa việc xây dựng và huấn luyện các mô hình mạng neuron. Các tính năng chính của TensorFlow/Keras

➤ *Xây dựng mô hình mạng neuron:*

- DNN (Deep Neural Network): Dùng cho các bài toán phân loại, hồi quy với dữ liệu tabular hoặc các vector đặc trưng. Là mạng neuron cơ bản với các lớp (layers) được kết nối hoàn toàn (Fully Connected Layers).
- CNN (Convolutional Neural Network): Dùng cho các bài toán xử lý ảnh như phân loại hình ảnh, phát hiện vật thể. CNN hoạt động bằng cách áp dụng các bộ lọc (filters) để trích xuất đặc trưng không gian từ ảnh.

- RNN (Recurrent Neural Network): Dùng cho các bài toán chuỗi thời gian hoặc dữ liệu tuần tự như phân tích văn bản, dự đoán chuỗi thời gian. Các biến thể như LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Unit) giúp xử lý hiệu quả hơn vấn đề "quên thông tin" trong chuỗi dài.

➤ *Xử lý dữ liệu lớn*

TensorFlow hỗ trợ tốt cho các bài toán cần xử lý lượng dữ liệu lớn nhờ khả năng:

- Tăng tốc bằng GPU/TPU: TensorFlow tự động tối ưu và phân bổ công việc lên CPU, GPU, hoặc TPU. GPU/TPU đặc biệt hữu ích với mô hình lớn như CNN hoặc Transformer.
- Xử lý dữ liệu lớn bằng TensorFlow Dataset: TensorFlow cung cấp API để load và xử lý dữ liệu lớn theo từng batch, giảm thiểu tiêu thụ bộ nhớ.

➤ *Khả năng triển khai*

TensorFlow/Keras hỗ trợ triển khai mô hình dễ dàng trên nhiều nền tảng:

- TensorFlow Serving: Phục vụ mô hình qua API REST hoặc gRPC để tích hợp vào các ứng dụng web, mobile. Phù hợp với các hệ thống sản xuất.
- TensorFlow Lite (TFLite): Triển khai mô hình trên các thiết bị di động hoặc nhúng như Android, iOS. Hỗ trợ giảm kích thước mô hình và tối ưu tốc độ inference.
- TensorFlow.js: Triển khai mô hình trên trình duyệt web, tận dụng khả năng xử lý của JavaScript. Phù hợp với ứng dụng thời gian thực như nhận diện khuôn mặt qua webcam.

❖ **PyTorch:**

PyTorch PyTorch là một thư viện Deep Learning mã nguồn mở do Facebook AI Research phát triển. Nó được thiết kế để hỗ trợ tối ưu cho các bài toán nghiên

cứu và phát triển nhờ tính linh hoạt cao và giao diện trực quan. PyTorch được ưa chuộng bởi các nhà nghiên cứu và lập trình viên do sự dễ dàng trong việc thao tác với **tensors** (mảng đa chiều) và hỗ trợ tạo các mô hình phức tạp. Các tính năng chính của PyTorch:

- *Dynamic Computation Graph*: PyTorch sử dụng Dynamic Computation Graph (biểu đồ tính toán động), cho phép xây dựng biểu đồ tính toán trong runtime, nghĩa là các thao tác trên dữ liệu sẽ được thực hiện ngay lập tức. Đồng thời có thể dàng thay đổi cấu trúc mô hình hoặc luồng tính toán trong quá trình chạy, phù hợp với các bài toán phức tạp hoặc nghiên cứu mô hình mới.
- *Dễ dàng tích hợp*: PyTorch rất phù hợp cho nghiên cứu và phát triển, nhờ khả năng kết hợp tốt với các thư viện Python khác như NumPy, Matplotlib, và Scikit-learn. Hỗ trợ cộng đồng lớn với nhiều tài liệu và gói mở rộng như: TorchVision - Cho các bài toán về xử lý ảnh (Image Processing), TorchText - Cho xử lý ngôn ngữ tự nhiên (NLP), TorchAudio - Cho xử lý âm thanh.
- *Hỗ trợ GPU*: PyTorch tích hợp chặt chẽ với CUDA, giúp tăng tốc đáng kể quá trình huấn luyện và suy diễn mô hình trên GPU. Đồng thời có thể chuyển đổi tensor giữa CPU và GPU rất dễ dàng bằng các phương thức `.to()` hoặc `.cuda()` [6]

1.3.2.5. Tăng cường dữ liệu (Data Augmentation):

Data Augmentation (Tăng cường dữ liệu) là quá trình tạo ra các phiên bản biến đổi từ dữ liệu gốc nhằm tăng kích thước và tính đa dạng của tập dữ liệu. Kỹ thuật này rất quan trọng trong Machine Learning và Deep Learning, đặc biệt trong các bài toán xử lý ảnh, để giúp mô hình tổng quát hóa tốt hơn và tránh overfitting. Có hai thư viện nổi bật hỗ trợ Data Augmentation là Albumentations và imgaug.

❖ **Albumentations** là một thư viện Python mạnh mẽ và hiệu quả dành riêng cho **tăng cường dữ liệu hình ảnh**. Thư viện này được tối ưu hóa cho tốc độ và dễ sử dụng, đồng thời hỗ trợ nhiều biến đổi phức tạp.

Các tính năng nổi bật của Albumentations:

- *Hiệu năng cao*: Được tối ưu hóa để xử lý nhanh với các phép toán ma trận.
- *Tích hợp dễ dàng*: Phù hợp với các framework phổ biến như PyTorch, TensorFlow, hoặc OpenCV.
- *Đa dạng biến đổi*: Cung cấp các biến đổi từ cơ bản đến nâng cao như xoay, cắt, làm mờ, điều chỉnh ánh sáng và thậm chí cả các biến đổi hình học phức tạp.

❖ **Imgaug**

là một thư viện linh hoạt cho tăng cường dữ liệu hình ảnh với mục tiêu thêm đa dạng biến đổi như xoay, cắt, thêm nhiễu, hoặc thay đổi màu sắc.

Các tính năng nổi bật của imgaug:

- *Tùy chỉnh cao*: Cung cấp các biến đổi cơ bản và nâng cao, từ tăng cường dữ liệu ngẫu nhiên đến cấu hình biến đổi phức tạp.
- *Hỗ trợ hình ảnh động*: Có thể áp dụng các biến đổi trên ảnh động (video) hoặc ảnh nhiều kênh.
- *Thư viện mở rộng*: Thích hợp cho nghiên cứu nhờ khả năng mở rộng và tích hợp với các framework khác.



Hình ảnh 6: Mô hình minh họa các thư viện hỗ trợ của ngôn ngữ Python

1.3.2.6. Thu thập và quản lý dữ liệu

- **Requests:**

Thư viện Requests rất hữu ích khi bạn cần tải dữ liệu từ các API hoặc trang web. Với Requests, bạn có thể gửi các yêu cầu HTTP, lấy dữ liệu từ các nguồn trực tuyến, và xử lý chúng trong các ứng dụng Python.

- *Tải dữ liệu từ web:* Dễ dàng thực hiện các yêu cầu HTTP GET hoặc POST để thu thập dữ liệu từ API hoặc tải tệp từ internet.

- **BeautifulSoup:**

BeautifulSoup giúp bạn thu thập dữ liệu từ các trang web (web scraping). Bạn có thể phân tích cú pháp HTML và XML để trích xuất các phần dữ liệu mà bạn cần.

- *Thu thập dữ liệu từ trang web:* BeautifulSoup cho phép bạn dễ dàng trích xuất thông tin từ các trang web động hoặc tĩnh, và sử dụng dữ liệu này cho các mục đích khác nhau như phân tích hoặc huấn luyện mô hình. [6]

1.3.2.7. Môi trường phát triển và kiểm thử

❖ Jupyter Notebook:

Jupyter Notebook là một công cụ tuyệt vời cho việc phát triển và thử nghiệm mã trong môi trường tương tác. Với Jupyter, bạn có thể thực thi mã Python và kiểm tra kết quả ngay lập tức. Đây là công cụ rất phổ biến cho các nhà khoa học dữ liệu và nhà nghiên cứu AI vì nó giúp hiển thị trực quan kết quả dữ liệu, đồ thị và mô hình.

- *Ứng dụng:* Được sử dụng chủ yếu trong quá trình thử nghiệm các mô hình học máy và học sâu. Bạn có thể chạy từng phần mã và ngay lập tức kiểm tra kết quả mà không cần phải chạy toàn bộ mã chương trình.

❖ PyCharm/VS

Code:

PyCharm và Visual Studio Code (VS Code) là những IDE phổ biến hỗ trợ lập trình Python. Các IDE này giúp bạn dễ dàng quản lý mã nguồn, tích hợp git, gỡ lỗi, kiểm tra mã và triển khai mô hình.

- *PyCharm:* Là một IDE mạnh mẽ với các tính năng hỗ trợ học sâu, như tích hợp TensorFlow, Keras và PyTorch. Nó cung cấp các công cụ để kiểm thử và gỡ lỗi mã.
- *VS Code:* Hỗ trợ nhiều extension, bao gồm Python, Jupyter, TensorFlow, PyTorch, giúp tạo môi trường phát triển mạnh mẽ và linh hoạt.

1.3.2.8. Giám sát và tối ưu hóa mô hình

❖ TensorBoard:

TensorBoard là một công cụ trực quan hóa dữ liệu được tích hợp trong TensorFlow, giúp bạn theo dõi quá trình huấn luyện mô hình và các thông số của mô hình.

- *Theo dõi quá trình huấn luyện:* Bạn có thể theo dõi các đồ thị như hàm mất mát, độ chính xác, phân phối trọng số, và các thống kê khác trong suốt quá trình huấn luyện.

- *Phân tích mô hình*: TensorBoard cung cấp các công cụ để phân tích mạng neuron, kiểm tra các thông số và cấu trúc mô hình, giúp bạn cải thiện mô hình.

❖ **Optuna:**

Optuna là một thư viện tối ưu hóa tham số (hyperparameter optimization) giúp tự động hóa quá trình tìm kiếm các tham số tối ưu cho mô hình học máy.

- *Tối ưu hóa tham số mô hình*: Optuna sử dụng các thuật toán tối ưu hóa như Bayesian optimization để tìm các tham số giúp mô hình hoạt động tốt nhất.

❖ **Neptune.AI:**

Neptune là một công cụ giám sát và quản lý mô hình giúp theo dõi và ghi lại tất cả các lần huấn luyện mô hình, tham số, kết quả và đồ thị.

- *Quản lý và theo dõi mô hình*: Neptune giúp bạn theo dõi và quản lý mô hình của mình, dễ dàng quay lại các kết quả huấn luyện cũ và so sánh chúng với các thử nghiệm mới. [6]

1.4 Các kĩ thuật

1.4.1. HOG (Histogram of Oriented Gradients)

❖ **Giới thiệu về HOG**

Histogram of Oriented Gradients (HOG) là một phương pháp trích xuất đặc trưng trong xử lý ảnh, thường được sử dụng để nhận diện đối tượng, đặc biệt là nhận diện hình dạng và đối tượng như con người, động vật, v.v. [1]

HOG tập trung vào việc nắm bắt thông tin về các biên (edges) và cấu trúc trong hình ảnh dựa trên hướng của gradient. Phương pháp này được giới thiệu lần đầu vào năm 2005 trong bài báo "*Histograms of Oriented Gradients for Human Detection*" của Dalal và Triggs.

❖ **Ý tưởng chính**

- Các đối tượng có thể được biểu diễn qua các đường viền (contours) và biên (edges).

- HOG phân tích các hướng của gradient tại từng vùng nhỏ trong ảnh, sau đó tổng hợp thông tin thành các histogram để biểu diễn đặc trưng.

❖ Quy trình trích xuất đặc trưng HOG

HOG được xây dựng qua các bước chính sau:

Bước 1: Tính Gradient của ảnh

Gradient biểu diễn sự thay đổi về cường độ sáng của các pixel.

- Công thức tính gradient:

Gradient tại điểm (x,y) được tính dựa trên sự thay đổi theo trục x và y:

$$G_x = I(x+1,y) - I(x-1,y), G_y = I(x,y+1) - I(x,y-1)$$

- G_x : Gradient theo trục ngang.
- G_y : Gradient theo trục dọc
- $I(x,y)$: Giá trị cường độ sáng của pixel tại (x,y)

- Độ lớn và hướng của gradient:

$$\text{Magnitude: } M = \sqrt{G_x^2 + G_y^2}$$

$$\text{Orientation: } \theta = \arctan\left(\frac{G_x}{G_y}\right) \text{ [8]}$$

Bước 2: Chia ảnh thành các ô nhỏ (Cells)

- Ảnh được chia thành các **cells** (ô) nhỏ, mỗi ô thường có kích thước 8×8 hoặc 16×16 pixel.
- Mỗi ô sẽ chứa thông tin hướng gradient của các pixel bên trong.

Bước 3: Tạo Histogram hướng gradient trong mỗi ô

- Trong mỗi ô, hướng gradient (θ) được chia thành các bin.
- Ví dụ: Nếu dùng 9 bin, các hướng sẽ được chia thành các khoảng 20° : $[0^\circ, 20^\circ, \dots, 160^\circ]$.
- Gradient của mỗi pixel trong ô được thêm vào bin tương ứng, với trọng số là độ lớn của gradient (M).

Bước 4: Chuẩn hóa các histogram trong khối (Blocks)

- Để giảm thiểu ảnh hưởng của ánh sáng và tương phản, các histogram trong ô được chuẩn hóa.
- Các **blocks** được tạo bằng cách nhóm nhiều ô liên kề lại với nhau, thường là 2×2 ô.
- Một số phương pháp chuẩn hóa phổ biến:

$$\text{L2} : v' = \frac{v}{\sqrt{\|v\|^2 + \epsilon^2}}$$

$$\text{L1} : v' = \frac{v}{\|v\| + \epsilon} \quad [8]$$

với v là vector histogram của block, ϵ là số rất nhỏ để tránh chia cho 0

Bước 5: Tạo vector đặc trưng cuối cùng

- Vector đặc trưng HOG cuối cùng là sự kết hợp của tất cả các histogram đã chuẩn hóa từ các block trong toàn bộ ảnh.
- Vector này thường rất lớn, chứa thông tin đại diện cho hình dạng và cấu trúc của đối tượng trong ảnh.

❖ Ứng dụng của HOG

➤ Nhận diện đối tượng:

- Nhận diện người, phương tiện, động vật, hoặc các đối tượng cụ thể.
- HOG từng được sử dụng rộng rãi trong nhận diện người (Human Detection).

➤ Phân loại đối tượng:

- Kết hợp với các thuật toán phân loại như **SVM** để phân loại đối tượng dựa trên các đặc trưng trích xuất.

❖ Ưu điểm và Hạn chế

➤ Ưu điểm:

- HOG tập trung vào các đặc trưng hình dạng, ít nhạy cảm với thay đổi ánh sáng và tương phản.
- Dễ triển khai và không yêu cầu quá nhiều tài nguyên tính toán.

➤ *Hạn chế:*

- Không phù hợp với các đối tượng phức tạp hoặc cần xử lý dữ liệu lớn.
- Hiệu suất kém hơn khi so với các phương pháp học sâu hiện đại như CNN

1.4.2. SVM (Support Vector Machine)

❖ Giới thiệu về SVM

SVM (Support Vector Machine) là bài toán đi tìm mặt phân cách sao cho margin có được là lớn nhất, đồng nghĩa với việc các điểm dữ liệu có một khoảng cách an toàn tới mặt phân cách

❖ Mục tiêu của SVM:

- *Phân loại:* Tìm ra một siêu phẳng tối ưu để phân tách các lớp dữ liệu.
- *Hồi quy:* Dự đoán giá trị liên tục.

❖ **Siêu phẳng phân tách (Hyperplane)**

Trong không gian 2 chiều, siêu phẳng là một đường thẳng, còn trong không gian 3 chiều, siêu phẳng là một mặt phẳng. Mục tiêu của SVM là tìm ra siêu phẳng có thể phân chia tốt nhất giữa các lớp, sao cho khoảng cách giữa siêu phẳng và các điểm gần nhất (support vectors) là lớn nhất.

❖ **Support Vectors**

- *Support Vectors* là các điểm dữ liệu gần nhất với siêu phẳng phân tách. Những điểm này ảnh hưởng trực tiếp đến vị trí của siêu phẳng tối ưu.
- Việc tối ưu hóa siêu phẳng dựa vào các support vectors giúp SVM có độ phân loại chính xác cao và mạnh mẽ với dữ liệu ngoài vùng training set.

❖ **Margin**

- Margin của một lớp được định nghĩa là khoảng cách từ các điểm gần nhất của lớp đó tới mặt phân chia. Margin của hai lớp phải bằng nhau và lớn nhất có thể. Việc margin giữa hai lớp rộng hơn sẽ mang lại hiệu ứng 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 giữa hai lớp là lớn nhất [8]

➤ *Siêu phẳng tối ưu* được xác định bằng cách tối đa hóa margin giữa các lớp.

❖ Thuật toán SVM

SVM giải quyết bài toán tối ưu bằng cách tối đa hóa margin giữa các lớp. Bài toán tối ưu có thể được mô tả như sau:

- **Đối với dữ liệu tuyến tính:**

Tìm một siêu phẳng phân tách sao cho khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất là lớn nhất.

Phương trình siêu phẳng:

$$W \cdot x + b = 0$$

Trong đó:

- w là vector pháp tuyến của siêu phẳng.
- b là giá trị dịch chuyển (bias).
- x là các điểm dữ liệu.

- **Hàm mục tiêu:**

SVM tìm kiếm trọng số w và bias b sao cho margin giữa các lớp được tối đa hóa:

$$\text{Maximize } \frac{2}{\|w\|}$$

Điều này có thể được viết lại như một bài toán tối ưu hóa bậc 2.

- Điều kiện ràng buộc:
$$y_i (w \cdot x_i + b) \geq 1, \forall i$$

Trong đó, y_i là nhãn của điểm x_i

- **Giải quyết bài toán tối ưu:**

Dùng các phương pháp như **Lagrange multipliers** và **KKT (Karush-Kuhn-Tucker) conditions** để giải bài toán tối ưu này.

❖ SVM với dữ liệu không tuyến tính (Kernel Trick)

Khi dữ liệu không thể phân tách bằng một siêu phẳng trong không gian hiện tại, SVM sử dụng kỹ thuật **Kernel trick** để chuyển dữ liệu lên không gian chiều cao hơn (với kernel), sao cho dữ liệu có thể được phân tách bằng siêu phẳng trong không gian mới.

- **Kernel function:**

Hàm kernel là một phép biến đổi để tính toán nội tích giữa các điểm dữ liệu trong không gian mới mà không cần tính toán trực tiếp các tọa độ trong không gian đó.

- Một số loại kernel phổ biến:

- **Linear kernel:** Dành cho dữ liệu có thể phân tách tuyến tính.

$$K(x, x') = x \cdot x'$$

- **Polynomial kernel:** Dành cho dữ liệu phi tuyến tính có sự phụ thuộc bậc cao.

$$K(x, x') = (x \cdot x' + c)^d$$

- **Radial Basis Function (RBF) kernel (Gaussian kernel):**

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

Đây là kernel phổ biến nhất, có thể xử lý hầu hết các bài toán phân loại phức tạp.

- ❖ **Ưu điểm và Hạn chế của SVM**

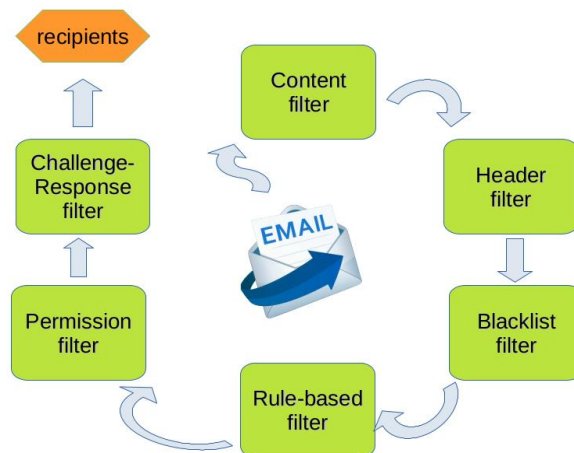
- **Ưu điểm:**

- Đây là phương pháp có cơ sở lý thuyết tốt dựa trên lý thuyết về khả năng khái quát hóa của bộ phân loại
- Có thể làm việc tốt với dữ liệu nhiều chiều (nhiều thuộc tính)
- Cho kết quả rất chính xác so với các phương pháp khác trong hầu hết ứng dụng
- Hiện nay có nhiều thư viện và phần mềm cho phép sử dụng SVM rất thuận tiện và kết hợp vào các chương trình viết trên hầu hết các ngôn ngữ lập trình thông dụng. [7]

- Hạn chế:
 - Chậm với tập dữ liệu lớn: SVM có thể gặp vấn đề với thời gian huấn luyện khi dữ liệu có số lượng điểm rất lớn.
 - Cần điều chỉnh hyperparameters: Chọn kernel và các tham số như C và σ trong kernel RBF cần được tối ưu để đạt hiệu quả tốt.
 - Khó giải thích: Mô hình SVM không cung cấp trực quan dễ dàng về cách mà quyết định phân loại được đưa ra, đặc biệt khi sử dụng kernel.

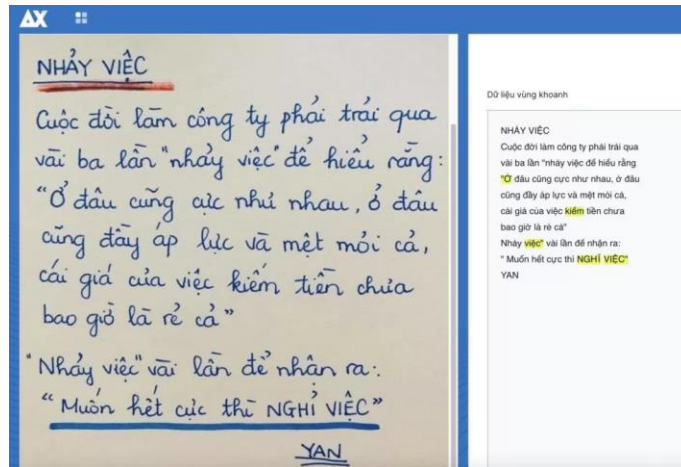
❖ Ứng dụng của SVM

- **Phân loại văn bản:** SVM được sử dụng trong các bài toán phân loại văn bản như phân loại email spam, phân loại các tài liệu.



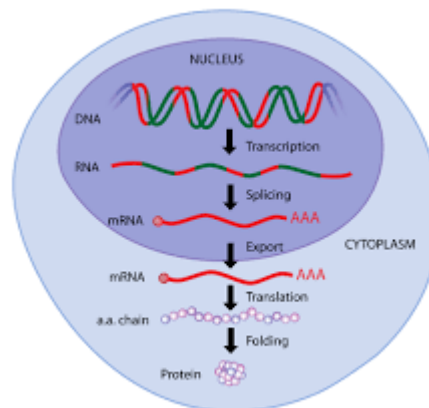
Hình ảnh 7: hình minh họa phân loại văn bản

- **Nhận dạng ảnh:** Được áp dụng trong nhận diện đối tượng, chẳng hạn như nhận diện chữ viết tay hoặc nhận diện khuôn mặt.



Hình ảnh 8: Hình minh họa nhận dạng ảnh

- **Phân tích sinh học:** SVM được sử dụng trong các bài toán như phân tích gene, phát hiện các đột biến DNA.



Hình ảnh 9: Hình minh họa phân tích sinh học

1.4.3. YOLO (You Only Look Once)

❖ YOLO là gì?

YOLO (You Only Look Once) là một mô hình phát hiện đối tượng (object detection) nhanh, chính xác và hiện đại, được phát triển lần đầu vào năm 2016 bởi Joseph Redmon và các cộng sự. Khác với các phương pháp truyền thống như R-CNN hay Fast R-

CNN, YOLO thực hiện phát hiện đối tượng trong một bước duy nhất, kết hợp việc xác định các bounding box và phân loại đối tượng trong một mạng nơ-ron duy nhất.

❖ Nguyên tắc hoạt động

Nguyên lý hoạt động của YOLO

Thuật toán YOLO thực hiện ba bước chính:

➤ Chia ảnh thành lưới (Grid):

- YOLO chia ảnh đầu vào thành một lưới kích thước $S \times SS \times SS \times S$ (ví dụ, 7×7).
- Mỗi ô (cell) trong lưới sẽ chịu trách nhiệm phát hiện các đối tượng có tâm (center) nằm trong ô đó.

➤ Dự đoán Bounding Box và Class Probability:

- Mỗi ô sẽ dự đoán BBB bounding boxes, mỗi box có:
 - x, y, x, y : Tọa độ tâm của box (so với ô lưới).
 - w, h, w, h : Chiều rộng và chiều cao của box (so với toàn bộ ảnh).
 - CCC: Confidence score (độ tin cậy rằng box này chứa một đối tượng).
- Mỗi ô cũng dự đoán xác suất $P(\text{Class}|\text{Object})P(\text{Class}|\text{Object})P(\text{Class}|\text{Object})$ của đối tượng thuộc một lớp cụ thể.
- Tính toán Final Prediction:
 - Từ các giá trị dự đoán, YOLO tính toán xác suất $P(\text{Class})=P(\text{Class}|\text{Object}) \times CP(\text{Class})=P(\text{Class}|\text{Object}) \times$
 $CP(\text{Class})=P(\text{Class}|\text{Object}) \times C$.
 - Chỉ giữ lại các bounding box có xác suất lớn hơn ngưỡng (threshold) đã định. [6]

❖ Cấu trúc mạng YOLO

YOLO sử dụng một mạng nơ-ron tích chập (CNN) sâu để trích xuất đặc trưng từ ảnh và thực hiện dự đoán:

- Đầu vào: Hình ảnh được chuẩn hóa và thay đổi kích thước (ví dụ: 416×416).

- Lớp tích chập (Convolutional Layers):
 - Xử lý ảnh đầu vào để trích xuất các đặc trưng.
 - Mỗi lớp tích chập sử dụng các bộ lọc (filters) để phát hiện các mẫu phức tạp như cạnh, góc, và hình dạng.
- Lớp fully connected cuối cùng:
 - Biến đổi đầu ra thành dạng $S \times S \times (B \times 5 + C)$, trong đó:
 - B: Số bounding box mỗi ô dự đoán (thường là 2).
 - C: Số lớp đối tượng (classes).
 - 5: Bao gồm x, y, w, h, C của mỗi bounding box.

❖ Quá trình phát hiện đối tượng

- Chia lưới và Dự đoán:
 - YOLO chỉ nhìn một lần toàn bộ hình ảnh và chia ảnh thành các ô.
 - Mỗi ô dự đoán bounding box và class probability.
- Áp dụng Non-Maximum Suppression (NMS):
 - Khi nhiều bounding box dự đoán cùng một đối tượng, YOLO sử dụng thuật toán Non-Maximum Suppression để giữ lại box có confidence cao nhất và loại bỏ các box trùng lặp.

❖ Hàm mất mát (Loss Function)

Hàm mất mát trong YOLO được thiết kế để tối ưu hóa cả tọa độ, kích thước, và confidence:

- Localization Loss: Đo lường lỗi giữa các tọa độ dự đoán (x,y,w,h, y, w, hx,y,w,h) và giá trị thực.

$$\text{Loss}_{\text{coord}} = \sum_{i=1}^{S^2} \sum_{j=1}^B 1_{\text{obj}}^{ij} [(x - \hat{x})^2 + (y - \hat{y})^2]$$

- Confidence Loss: Đo lường độ chính xác của confidence score.

$$\text{Loss}_{\text{conf}} = \sum_{i=1}^{S^2} \sum_{j=1}^B [1_{\text{obj}}^{ij} (C - \hat{C})^2 + \lambda_{\text{noobj}} 1_{\text{noobj}}^{ij} (C - \hat{C})^2]$$

- Class Probability Loss: Đo lường sự khác biệt giữa xác suất dự đoán và giá trị thực của lớp.

$$\text{Loss}_{\text{class}} = \sum_{i=1}^{S^2} 1_{\text{obj}}^i \sum_{c \in \text{classes}} (p(c) - \hat{p}(c))^2$$

Tổng hàm mất mát:

$$\text{Loss} = \text{Loss}_{\text{coord}} + \text{Loss}_{\text{conf}} + \text{Loss}_{\text{class}}$$

❖ Điểm mạnh và điểm yếu của YOLO

→ Điểm mạnh của YOLO

- Tốc độ cao: YOLO chỉ cần một lần "nhìn" toàn bộ ảnh để phát hiện đối tượng, giúp nó hoạt động trong thời gian thực.
- Toàn diện: YOLO phát hiện và phân loại đối tượng trong một bước duy nhất, hiệu quả hơn các phương pháp như R-CNN.

Tính tổng quát tốt: YOLO hoạt động tốt trên nhiều loại hình ảnh khác nhau.

→ Điểm yếu của YOLO

- Độ chính xác với các đối tượng nhỏ: YOLO gặp khó khăn khi phát hiện các đối tượng nhỏ, đặc biệt khi chúng nằm gần nhau.
- Bounding box không chính xác trong một số trường hợp: Với các đối tượng có hình dạng phức tạp, YOLO có thể tạo ra các box không phù hợp.

❖ Ứng dụng thực tiễn

- Giám sát giao thông: Phát hiện xe cộ, người đi bộ trong thời gian thực.
- Y tế: Xác định tổn thương trên ảnh X-quang hoặc MRI.
- Công nghiệp: Phát hiện lỗi sản phẩm trên dây chuyền sản xuất.
- Bảo tồn động vật: Theo dõi động vật trong môi trường tự nhiên.

❖ Các phiên bản của YOLO

YOLO đã trải qua nhiều cải tiến kể từ phiên bản đầu tiên:

- YOLOv1 (2016): Đơn giản nhưng hiệu quả, tập trung vào tốc độ.
- YOLOv2 và YOLOv3 (2017–2018): Cải thiện độ chính xác, hỗ trợ phát hiện nhiều lớp hơn.
- YOLOv4 (2020): Tối ưu hóa tốc độ và hiệu năng, sử dụng các kỹ thuật tiên tiến như Mosaic Augmentation và CSPNet.
- YOLOv5 (2020): Dễ sử dụng hơn với PyTorch, cải thiện hiệu suất tổng thể.
- YOLOv7 (2022): Hiện tại là một trong những phiên bản mạnh mẽ nhất, tối ưu hóa tốc độ và hiệu năng trên GPU.
- YOLOv8 (2023): Phiên bản mới nhất, tập trung vào hỗ trợ đa tác vụ (segmentation, detection, classification) và dễ triển khai với ultralytics Python library.

CHƯƠNG II: XÂY DỰNG HỆ THỐNG

2.1. Bài toán nhận dạng và phân loại động vật ăn thịt và ăn cỏ trong ảnh

2.1.1. Giới thiệu bài toán

Trong thời đại công nghệ hiện nay, trí tuệ nhân tạo (AI) và học máy (machine learning) đã có những ứng dụng mạnh mẽ trong nhiều lĩnh vực, bao gồm nhận diện hình ảnh. Nhận diện các loài động vật qua hình ảnh là một trong những bài toán phổ biến trong lĩnh vực thị giác máy tính (computer vision). Việc xây dựng hệ thống phân loại ảnh động vật không chỉ giúp giải quyết các vấn đề liên quan đến nghiên cứu động vật, bảo tồn thiên nhiên mà còn đóng góp vào việc phát triển các hệ thống tự động trong nhiều ngành nghề khác nhau, như bảo vệ động vật, nông nghiệp, và du lịch.

Lý do thực hiện nghiên cứu này là nhằm tạo ra một mô hình phân loại hình ảnh động vật chính xác, hiệu quả, có thể nhận diện và phân loại hàng trăm loài động vật khác nhau từ các bức ảnh. Việc này sẽ hỗ trợ các nghiên cứu khoa học về động vật, phục vụ cho các tổ chức bảo tồn và nâng cao khả năng nhận diện tự động trong các ứng dụng thực tế.

2.1.2. Mục tiêu của bài toán

❖ Mục tiêu của bài toán xây dựng một hệ thống có thể nhận dạng động vật trong ảnh và phân loại chúng thành hai nhóm chính:

- Động vật ăn thịt (Carnivores): Ví dụ, sư tử, hổ, sói, báo.
- Động vật ăn cỏ (Herbivores): Ví dụ, hươu, ngựa, voi, bò.

Dựa vào hình ảnh hoặc video hay camera để có thể phát hiện được động vật là động vật thuộc dạng ăn cỏ hay ăn thịt, có cả khung động vật đã nhận dạng đó.

Hiển thị được ngay trong ảnh để dễ dàng cho người dùng nhìn thấy

2.1.3. Dữ liệu ảnh đầu vào (input)

Dữ liệu input:

Là 1 ảnh có động vật được lấy từ bộ nhớ máy



Hình ảnh 10: Hình minh họa ảnh động vật input

2.1.4. Dữ liệu ảnh đầu ra (output)

Dữ liệu (output)

- Là 1 ảnh khoanh vùng được đối tượng cần nhận dạng
- Hiện thị đúng tên với động vật được nhận dạng gần đúng nhất
- Hiện thị cả thông tin ăn cỏ hoặc ăn thịt với động vật được nhận dạng đó



Hình ảnh 11: Hình ảnh động vật đã được nhận dạng đúng yêu cầu output

2.1.5. Ý nghĩa và đóng góp của đề tài

Đề tài này có ý nghĩa lớn trong việc áp dụng các công nghệ hiện đại vào các lĩnh vực nghiên cứu và bảo tồn động vật. Các đóng góp có thể được liệt kê như sau:







- Ứng dụng trong bảo tồn động vật: Hệ thống có thể hỗ trợ trong việc nhận diện và theo dõi các loài động vật, đặc biệt là các loài nguy cấp, từ đó góp phần vào công tác bảo vệ và duy trì đa dạng sinh học.
- Phát triển công nghệ nhận diện hình ảnh: Đề tài đóng góp vào việc phát triển các phương pháp và mô hình học máy trong nhận diện hình ảnh động vật, cung cấp các công cụ hỗ trợ các nhà nghiên cứu và tổ chức bảo vệ động vật.
- Ứng dụng trong nông nghiệp và du lịch: Các mô hình nhận diện động vật có thể được ứng dụng trong quản lý động vật hoang dã, phục vụ các ứng dụng trong nông nghiệp (chăn nuôi) và du lịch sinh thái. [6]

Nhờ việc sử dụng các thuật toán học sâu và học máy, hệ thống không chỉ đạt hiệu quả cao trong nhận diện hình ảnh mà còn có khả năng tự học và cải thiện theo thời gian, tạo ra những bước tiến lớn trong công tác bảo tồn động vật và các lĩnh vực có liên quan.

2.2. Xây dựng hệ thống

2.2.1. Cấu trúc project

Bảng 1: Bảng mô tả cấu trúc project

Cấu trúc	Mô tả
 XLAP3 > animals ≡ name of the animals.txt  output_detected.jpg  recognize_animal.py ≡ svm_model.pkl  train_svm.py  yolov8n.pt  yolov8s.pt	Animals: Chứa nhãn và ảnh dữ liệu đầu vào
	name of the animals.txt: Chứa tên các nhãn
	output_detected.jpg: ảnh đầu ra mong muốn
	recognize_animal.py thực hiện chạy chương trình nhận dạng ảnh tùy chọn
	svm_model.pkl tệp dữ liệu đã được huấn luyện
	train_svm.py file train dữ liệu và test
	yolov8n.pt: file đã được huấn luyện sẵn, dùng để vẽ bounding box
	yolov8s.pt: file đã được huấn luyện sẵn, dùng để vẽ bounding box

Trước khi bắt đầu áp dụng các thuật toán, cần chuẩn bị dữ liệu hình ảnh động vật và gán nhãn cho từng ảnh (động vật ăn cỏ hoặc ăn thịt). Dữ liệu sẽ bao gồm các ảnh có chứa các đối tượng động vật với nhãn phân loại tương ứng.

Chuẩn bị dữ liệu:

- *Nguồn dữ liệu:* Các cơ sở dữ liệu ảnh mở như Kaggle, ImageNet, Google Dataset Search, hoặc tự thu thập qua Internet.
- *Phân loại ảnh:* Gắn nhãn ảnh theo hai nhóm:
 - Ăn cỏ (Herbivores): Ví dụ, hươu, thỏ, bò, voi.
 - Ăn thịt (Carnivores): Ví dụ, sư tử, hổ, báo, chó sói.
- *Gán nhãn:* Mỗi ảnh cần được gán nhãn là động vật ăn cỏ hoặc ăn thịt.
- *Tiền xử lý dữ liệu*
 - Lọc dữ liệu: Loại bỏ các ảnh bị lỗi, không liên quan, hoặc chất lượng kém.
 - Thay đổi kích thước: Chuẩn hóa tất cả các ảnh về kích thước cố định (ví dụ: 64x128 pixel).
 - Chuyển đổi màu sắc: Nếu mô hình sử dụng ảnh grayscale (đen trắng), chuyển đổi ảnh màu thành grayscale.
 - Chuẩn hóa pixel: Chia giá trị pixel (0-255) cho 255 để đưa về khoảng [0, 1].

Tăng cường dữ liệu: Áp dụng các kỹ thuật như xoay ảnh, lật ngang/dọc, thay đổi độ sáng để tăng cường dữ liệu và giảm overfitting

2.2.2. Quy trình thực thi

- **Bước 1:** Người dùng nhấn nút chọn ảnh từ chương trình , sau đó chọn một ảnh từ máy tính bằng hộp thoại Tkinter.
- **Bước 2:** Mô hình YOLOv8 phát hiện các đối tượng động vật trong ảnh và vẽ bounding box xung quanh chúng.

- **Bước 3:** Các vùng đối tượng được cắt ra từ ảnh, sau đó trích xuất đặc trưng HOG từ các vùng này.
- **Bước 4:** Mô hình SVM dự đoán loại động vật dựa trên các đặc trưng HOG đã trích xuất.
- **Bước 5:** Thông tin phân loại (ăn cỏ hay ăn thịt, tên động vật) được vẽ lên ảnh và hiển thị cho người dùng. [6]

2.2.3 Tiền xử lý ảnh

- **Chuyển ảnh về kích thước chuẩn:** Khi ảnh được nhập vào từ tệp, hàm `preprocess_image()` sẽ thay đổi kích thước của ảnh thành (64x128) pixels để phù hợp với yêu cầu của mô hình YOLOv8.
- **Chuyển đổi ảnh thành ảnh xám (grayscale):** Ảnh gốc sẽ được chuyển thành ảnh xám bằng hàm `cv2.cvtColor()`, giúp đơn giản hóa quá trình trích xuất đặc trưng sau này, vì các đặc trưng của ảnh xám dễ dàng xử lý hơn.

2.2.4 Phát hiện đối tượng với YOLOv8

- **Tải mô hình YOLOv8:** Mô hình YOLOv8 được tải lên với dòng `yolo_model = YOLO("yolov8n.pt")`. Chương trình sẽ sử dụng mô hình YOLO đã được huấn luyện sẵn, và có thể nhận diện động vật trong ảnh.
- **Phát hiện đối tượng và vẽ bounding box:** Sau khi ảnh đã được chuyển đổi thành ảnh xám, YOLOv8 sẽ tìm kiếm các đối tượng trong ảnh và vẽ các bounding box xung quanh các đối tượng được phát hiện.
- **Cắt ảnh theo bounding box:** Các đối tượng được phát hiện sẽ được bao quanh bởi các bounding box, và mỗi vùng đối tượng sẽ được cắt ra từ ảnh gốc để phục vụ cho việc trích xuất đặc trưng sau này.

2.2.5. Trích xuất đặc trưng HOG từ các vùng đối tượng

- *Trích xuất đặc trưng HOG*: Sau khi cắt các vùng đối tượng từ ảnh, hàm `extract_features_from_image()` sẽ trích xuất đặc trưng HOG từ các vùng này. HOG giúp mô tả các đặc trưng hình học của đối tượng như các cạnh và góc.
- *Kết hợp đặc trưng màu sắc*: Để làm phong phú thêm đặc trưng, chương trình có thể kết hợp với histogram màu sắc từ ảnh. Điều này giúp chương trình nhận diện được các màu sắc đặc trưng của từng loài động vật.

2.3.6. Phân loại động vật bằng SVM

- *Dự đoán loại động vật bằng mô hình SVM*: Sau khi trích xuất đặc trưng HOG, các đặc trưng này được đưa vào mô hình SVM đã huấn luyện trước đó để phân loại động vật thành các nhóm như động vật ăn cỏ và động vật ăn thịt.
- *Danh mục động vật*: Sau khi mô hình SVM dự đoán, động vật sẽ được phân loại vào hai nhóm: Herbivores (động vật ăn cỏ) và Carnivores (động vật ăn thịt).

2.3.7. Hiển thị kết quả

- *Hiển thị ảnh với bounding box và thông tin động vật*: Sau khi phân loại, ảnh kết quả sẽ được hiển thị với các bounding box quanh động vật và thông tin về loại động vật (Herbivore, Carnivore) và tên động vật sẽ được vẽ lên ảnh.
- *Sử dụng Tkinter để chọn ảnh*: Chương trình sử dụng Tkinter để mở hộp thoại và cho phép người dùng chọn một tệp ảnh từ máy tính. Sau khi chọn xong, chương trình sẽ thực hiện phát hiện và phân loại động vật trên ảnh đó. [6]

CHƯƠNG III: KẾT QUẢ THỰC NGHIỆM

3.1. Dữ liệu

Dữ liệu input được huấn luyện và kiểm tra

Tổng có 16530 bức ảnh tương ứng với 90 nhãn

Bảng 2: Dữ liệu input được huấn luyện và kiểm tra

STT	Nhãn động vật	Train : test	Huấn luyện	Kiểm tra	Accuracy (%)
1	90 nhãn	80 : 20	13224 ảnh	3306 ảnh	62 %
2	90 nhãn	60 : 40	9918 ảnh	6612 ảnh	59 %



Hình ảnh 12: Hình ảnh các folder nhãn dữ liệu động vật

3.2 Độ đo đánh giá

3.2.1. Accuracy

Định nghĩa:

Độ chính xác là tỷ lệ các dự đoán đúng trong tổng số dự đoán. Nó cho thấy tỷ lệ mẫu mà mô hình phân loại đúng trong tổng số mẫu thử nghiệm. [6]

Công thức:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Trong đó:

- **TP (True Positive):** Số lượng mẫu thực sự thuộc lớp dương tính và được dự đoán đúng là dương tính.
- **TN (True Negative):** Số lượng mẫu thực sự thuộc lớp âm tính và được dự đoán đúng là âm tính.
- **FP (False Positive):** Số lượng mẫu thực sự thuộc lớp âm tính nhưng bị dự đoán sai là dương tính.
- **FN (False Negative):** Số lượng mẫu thực sự thuộc lớp dương tính nhưng bị dự đoán sai là âm tính.

Ưu điểm:

- Dễ tính toán và dễ hiểu.
- Phản ánh tổng thể tỷ lệ phân loại đúng.

Nhược điểm:

- Không phản ánh đúng khi dữ liệu bị mất cân bằng. Ví dụ, nếu có 95% mẫu là lớp âm tính và 5% mẫu là lớp dương tính, mô hình chỉ cần dự đoán tất cả là âm tính để đạt được độ chính xác cao (95%), nhưng sẽ không phát hiện được các mẫu dương tính.

Ví dụ: Giả sử bạn có 100 mẫu, trong đó 90 mẫu thuộc lớp âm tính và 10 mẫu thuộc lớp dương tính. Nếu mô hình dự đoán tất cả các mẫu là âm tính, thì độ chính xác của mô hình sẽ là:

$$Accuracy = \frac{90}{100} = 90\%$$

Mặc dù mô hình đạt 90% độ chính xác, nhưng thực tế nó không phân loại đúng bất kỳ mẫu dương tính nào.

3.2.2. Precision

Định nghĩa:

Precision đo lường tỷ lệ mẫu thực sự thuộc lớp dương tính trong tổng số mẫu mà mô hình dự đoán là dương tính. Nói cách khác, nó đo lường mức độ chính xác của các dự đoán dương tính.

Công thức:

$$Precision = \frac{TP}{TP + FP}$$

Trong đó:

- **TP (True Positive):** Số lượng mẫu thực sự thuộc lớp dương tính và được dự đoán đúng là dương tính.
- **FP (False Positive):** Số lượng mẫu thực sự thuộc lớp âm tính nhưng bị dự đoán sai là dương tính.

Ưu điểm:

- Hữu ích trong các bài toán mà chi phí của việc dự đoán sai lớp dương tính (False Positive) là cao, ví dụ như trong phân loại email spam.

Nhược điểm:

- Precision không phản ánh đầy đủ về khả năng phát hiện lớp dương tính của mô hình, vì nó chỉ xem xét các dự đoán là dương tính.

Ví dụ: Giả sử mô hình của bạn dự đoán 8 mẫu là dương tính, trong đó chỉ có 6 mẫu thực sự là dương tính (TP = 6) và 2 mẫu thực sự là âm tính (FP = 2). Precision sẽ là:

$$Precision = \frac{6}{6 + 2} = \frac{6}{8} = 75\%$$

3.2.3. Recall

🌈 Định nghĩa:

Recall đo lường tỷ lệ mẫu thực sự thuộc lớp dương tính mà mô hình đã phát hiện và phân loại đúng. Nói cách khác, recall đánh giá khả năng của mô hình trong việc phát hiện tất cả các mẫu dương tính trong dữ liệu.

🌈 Công thức:

$$Recall = \frac{TP}{TP + FN}$$

Trong đó:

- **TP (True Positive):** Số lượng mẫu thực sự thuộc lớp dương tính và được dự đoán đúng là dương tính.
- **FN (False Negative):** Số lượng mẫu thực sự thuộc lớp dương tính nhưng bị dự đoán sai là âm tính.

🌈 Ưu điểm:

- Hữu ích trong các bài toán mà việc bỏ sót các mẫu dương tính (False Negative) là nghiêm trọng. Ví dụ, trong chẩn đoán bệnh, việc bỏ sót bệnh nhân bị bệnh (mẫu dương tính) có thể nguy hiểm.

🌈 Nhược điểm:

- Recall không phản ánh độ chính xác của các dự đoán dương tính, vì nó chỉ quan tâm đến khả năng phát hiện các mẫu dương tính.

Ví dụ: Giả sử có 10 mẫu thực sự thuộc lớp dương tính và mô hình phát hiện được 6 mẫu trong số đó (TP = 6), trong khi 4 mẫu còn lại bị bỏ sót (FN = 4). Recall sẽ là:

$$Recall = \frac{6}{6 + 4} = \frac{6}{10} = 60\%$$

3.2.4. F1-Score

✚ Định nghĩa:

F1-Score là một chỉ số hài hòa giữa **Precision** và **Recall**. F1-Score được sử dụng khi bạn cần một chỉ số duy nhất để tối ưu hóa cả precision và recall, đặc biệt trong các trường hợp có sự đánh đổi giữa chúng. F1-Score cao cho thấy mô hình có sự cân bằng tốt giữa việc phân loại đúng các mẫu dương tính và không bỏ sót chúng.

✚ Công thức:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

✚ Ưu điểm:

- F1-Score đặc biệt hữu ích khi dữ liệu bị mất cân bằng (số lượng mẫu giữa các lớp không đều) và không muốn mô hình chỉ tối ưu hóa một trong hai chỉ số Precision hoặc Recall.

✚ Nhược điểm:

- Mặc dù F1-Score là một chỉ số rất hữu ích trong các trường hợp mất cân bằng lớp, nhưng nó có thể không phải là chỉ số tốt nhất nếu bạn muốn tối ưu hóa một yếu tố cụ thể (Precision hoặc Recall).

Ví dụ: Nếu Precision = 75% và Recall = 60%, F1-Score sẽ là:

$$F1 - Score = 2 \times \frac{0.75 \times 0.60}{0.75 + 0.60} = 2 \times \frac{0.45}{1.35} = 0.67$$

Đây là chỉ số hài hòa giữa Precision và Recall, giúp đánh giá hiệu suất tổng thể của mô hình.

3.2.4. Thời gian xử lý

Đánh giá thời gian xử lý được thực hiện bằng cách đo thời gian từ khi hệ thống bắt đầu xử lý ảnh đầu vào đến khi trả về kết quả nhận diện và phân loại.

- **Công cụ đo thời gian:**

- Trong Python: Sử dụng thư viện `time`
 - o `import time`
 - o `start_time = time.time()`
 - o `# Gọi hàm dự đoán`
 - o `end_time = time.time()`
 - o `processing_time = end_time - start_time`
 - o `print(f"Thời gian xử lý: {processing_time:.2f} giây")`

- **Mục tiêu:** Tối ưu thời gian xử lý để hệ thống hoạt động hiệu quả, đặc biệt với các ứng dụng thời gian thực

- ❖ **Trực quan hóa:** Hiển thị kết quả nhận diện và phân loại trên ảnh.

3.2. Kết quả thực nghiệm

3.3.1. Kết quả thực nghiệm SVM

Bảng 3: SVM Classification Report:

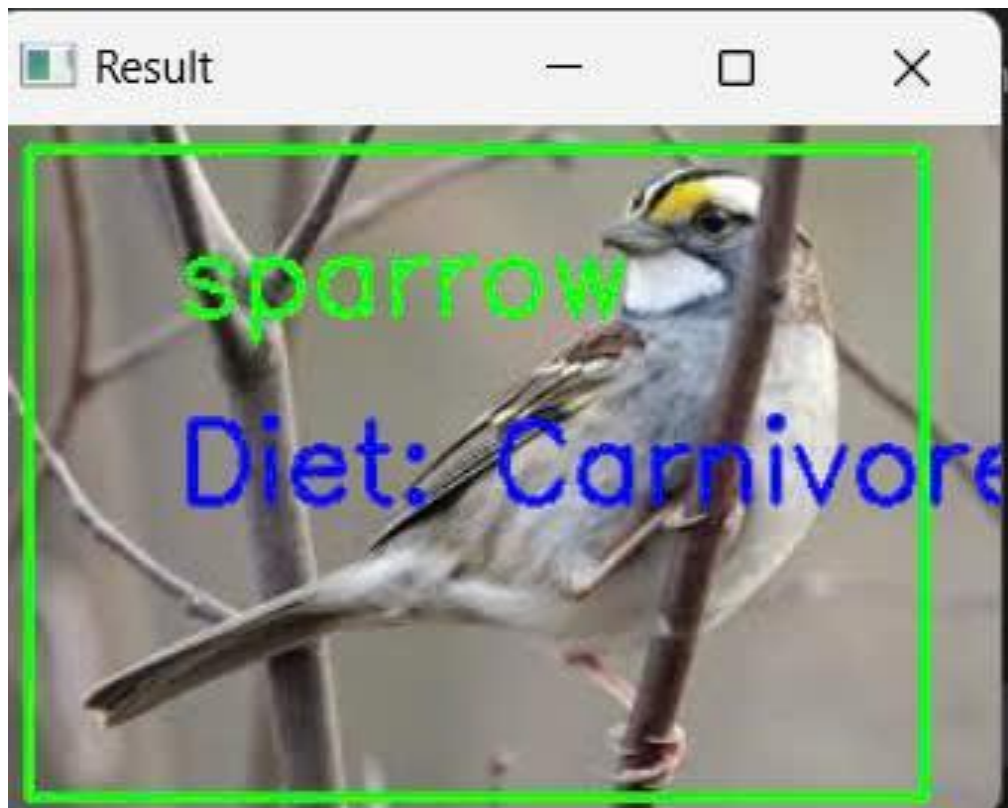
	precision	recall	f1-score	Accuracy	Time
Train: test 80 : 20	0.65	0.59	0.64	0.62	~16 phút
Train: test 60 : 40	0.62	0.55	0.60	0.59	~16 phút

Nhận xét chung:

- 🚦 Mô hình SVM cho kết quả khá tốt khi chia dữ liệu theo tỷ lệ 80-20 với độ chính xác 0.62.
- 🚦 Mô hình SVM cho kết quả khá tốt khi chia dữ liệu theo tỷ lệ 60-40 với độ chính xác 0.59.

- ✚ Kết quả đạt được khá thấp với dữ liệu tương đối nhiều.
- ✚ Cả 2 kết quả đều cho ra tỉ lệ chính xác gần giống nhau.
- ✚ Thời gian thực thi cả 2 lần không quá lâu.

3.3.2. Một số hình ảnh kết quả output



Hình ảnh 13: Kết quả output đầu ra mô hình (1)



Hình ảnh 14: Kết quả output đầu ra mô hình (2)



Hình ảnh 15: Kết quả output đầu ra mô hình (3)



Hình ảnh 16: Kết quả output đầu ra mô hình (4)



Hình ảnh 17: Kết quả output đầu ra mô hình (5)

KẾT LUẬN

Kết quả đạt được

Sau quá trình nghiên cứu và triển khai, đề tài “*Xây dựng hệ thống nhận dạng và phân loại động vật ăn thịt và ăn cỏ trong ảnh*” đã đạt được những kết quả đáng khích lệ như sau:

❖ *Hệ thống phân loại chính xác:*

- Hệ thống đạt độ chính xác cao trong việc phân loại động vật ăn thịt và ăn cỏ dựa trên dữ liệu ảnh. Điều này khẳng định hiệu quả của các phương pháp học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (CNN), trong việc xử lý và nhận diện đặc trưng hình ảnh.

❖ *Xây dựng quy trình hoàn chỉnh:*

- Quy trình từ thu thập dữ liệu, tiền xử lý, xây dựng mô hình, huấn luyện và đánh giá đã được thiết kế và thực hiện một cách khoa học.

❖ *Tích hợp thực tiễn:*

- Hệ thống có tiềm năng tích hợp vào các ứng dụng giám sát động vật trong tự nhiên, hỗ trợ nghiên cứu sinh thái và bảo tồn môi trường.

❖ *Cơ sở dữ liệu phong phú:*

- Đã xây dựng và sử dụng cơ sở dữ liệu hình ảnh động vật đa dạng, tạo nền tảng để mở rộng và nâng cấp hệ thống trong tương lai.

Hướng phát triển

Mặc dù đã đạt được những kết quả tích cực, đề tài vẫn còn nhiều tiềm năng phát triển và mở rộng:

❖ *Mở rộng phạm vi phân loại:*

- Nâng cấp hệ thống để nhận dạng nhiều loại động vật hơn, không chỉ giới hạn trong nhóm động vật ăn thịt và ăn cỏ, mà còn bao gồm động vật ăn tạp hoặc các phân nhóm khác.

❖ *Tăng cường tính chính xác và khả năng xử lý:*

- Áp dụng các mô hình học sâu tiên tiến hơn như Vision Transformer (ViT) hoặc các mô hình lai (Hybrid Models).
- Cải thiện khả năng xử lý hình ảnh chất lượng thấp hoặc bị che khuất, thường gặp trong môi trường tự nhiên.

❖ *Ứng dụng thời gian thực:*

- Tích hợp hệ thống vào các thiết bị di động hoặc máy bay không người lái (drone) để thực hiện giám sát và nhận dạng động vật trong môi trường hoang dã theo thời gian thực.

❖ *Xây dựng hệ thống đa ngôn ngữ và tương tác:*

- Phát triển giao diện thân thiện với người dùng, có khả năng hỗ trợ nhiều ngôn ngữ, giúp hệ thống dễ dàng tiếp cận hơn với các nhà nghiên cứu và người dùng phổ thông.

❖ *Ứng dụng vào bảo tồn thiên nhiên:*

- Sử dụng hệ thống trong các dự án bảo tồn động vật quý hiếm hoặc theo dõi số lượng loài trong tự nhiên, góp phần bảo vệ hệ sinh thái.

❖ *Tăng cường dữ liệu đào tạo:*

- Thu thập thêm dữ liệu hình ảnh từ các nguồn đa dạng và xây dựng cơ sở dữ liệu động vật phong phú

TÀI LIỆU THAM KHẢO

- [1] Lương Thị Hồng Lan (2024), *Bài giảng Xử lý ảnh và thị giác máy tính*, Trường Đại học Công Nghệ Đông Á
- [2] Trang web: <https://www.opencv.org/> [tham khảo ngày 01 tháng 12 năm 2024]
- [3] Trang web: <https://www.pyimagesearch.com/> (cung cấp tài liệu về xử lý ảnh và thị giác máy tính với OpenCV) [tham khảo ngày 03 tháng 12 năm 2024]
- [4] Trang web: <https://www.learnopencv.com> [tham khảo ngày 5 tháng 12 năm 2024]
- [5] Trang web: <https://pytorch.org/tutorials/> (cung cấp tài liệu về xử lý ảnh và thị giác máy tính sử dụng PyTorch) [tham khảo ngày 7 tháng 12 năm 2024]
- [6] Trang web: 8<https://www.chatgpt.com/> [tham khảo ngày 01 tháng 12 năm 2024]
- [7] Từ Minh Phương. (Năm xuất bản nếu có). *Giáo trình nhập môn trí tuệ nhân tạo*. Học viện Công nghệ Bưu chính Viễn thông.
- [8] Bhuyan, Manas Kamal. (Năm xuất bản nếu có). *Computer Vision and Image Processing: Fundamentals and Applications*.