

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN CUỐI KỲ
NHẬP MÔN THỊ GIÁC MÁY TÍNH - CS231

ĐỀ TÀI: ANIMALS CLASSIFICATION
(Link Github: CS231-ComputerVision)

GV hướng dẫn: TS. Mai Tiến Dũng

Nhóm sinh viên thực hiện:

STT	Họ và tên	MSSV
1	Trần Như Cẩm Nguyên	22520004
2	Trần Thị Cẩm Giang	22520361

TP. Hồ Chí Minh, Tháng 6 năm 2024

LỜI CẢM ƠN

Nhóm xin chân thành gửi lời cảm ơn đến **TS. Mai Tiến Dũng** – Giảng viên khoa Khoa học máy tính, Trường Đại học Công nghệ Thông tin, Đại học Quốc gia thành phố Hồ Chí Minh, đồng thời là giảng viên giảng dạy lớp CS231.O21.KHTN – Môn Thị giác máy tính, trong thời gian đã tận tình hướng dẫn và định hướng cho nhóm trong suốt quá trình thực hiện và hoàn thành đồ án.

Trong quá trình thực hiện đồ án, nhóm đã cố gắng rất nhiều để hoàn thành đồ án một cách tốt nhất và hoàn thiện nhất, tuy nhiên cũng không tránh khỏi được những sai sót ngoài ý muốn. Nhóm mong rằng sẽ nhận được những lời nhận xét và những lời góp ý chân thành từ quý thầy và các bạn trong quá trình để hoàn thiện đồ án hơn. Mọi thắc mắc cũng như mọi góp ý của mọi người xin gửi email về một trong các địa chỉ email sau: **222520004@gm.uit.edu.vn** (Trần Như Cẩm Nguyên) hoặc **22520361@gm.uit.edu.vn** (Trần Thị Cẩm Giang).

Chúng em xin chân thành cảm ơn!

Đánh giá mức độ hoàn thành

Họ và tên	MSSV	Công việc thực hiện	Mức độ hoàn thành
Trần Như Cẩm Nguyên	22520004	<ul style="list-style-type: none"> • Lên ý tưởng • Chuẩn bị nội dung • Thực nghiệm với KNN và RF • Slide Thuyết trình • Viết báo cáo 	100%
Trần Thị Cẩm Giang	22520361	<ul style="list-style-type: none"> • Lên ý tưởng • Chuẩn bị nội dung • Thực nghiệm với SVM • Thống kê và chuẩn bị demo • Viết báo cáo 	100%

Mục lục

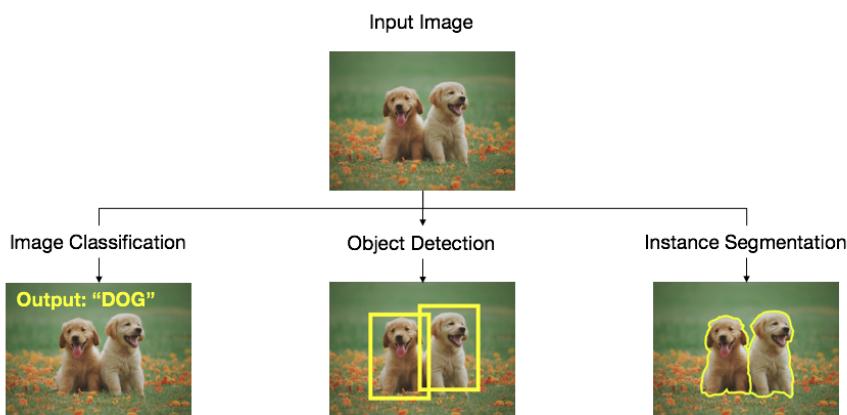
1 Giới thiệu đề tài	3
1.1 Tổng quan	3
1.2 Phát biểu bài toán	4
1.3 Nội dung thực hiện	4
2 Cơ sở lý thuyết	5
2.1 Histogram Of Oriented Gradient - HOG	5
2.2 Principal Component Analysis - PCA	9
2.3 Support Vector Machine - SVM	9
2.4 k-Nearest Neighbors - KNN	12
2.5 Random Forest - RF	14
3 Thực nghiệm	16
3.1 Dataset	16
3.2 Metrics	17
3.3 Cài đặt	18
3.4 Đánh giá	18
4 Nhận xét và kết luận	20
4.1 Ưu điểm	20
4.2 Hạn chế	21
4.3 Hướng phát triển	21
5 Tài liệu tham khảo	21

1 Giới thiệu đề tài

1.1 Tổng quan

Thị giác máy tính (Computer Vision) là một nhánh trong ngành khoa học máy tính, cho phép máy tính và hệ thống lấy thông tin có ý nghĩa từ hình ảnh kỹ thuật số, video và các đầu vào trực quan khác. Từ đó, máy tính có thể thực hiện hành động hoặc đưa ra đề xuất dựa trên thông tin đó. Nếu trí tuệ nhân tạo cho phép máy tính suy nghĩ, thì thị giác máy tính cho phép máy tính nhìn, quan sát và hiểu được hình ảnh.

Phân loại ảnh (Image Classification) là một trong những nhiệm vụ phổ biến trong xử lý dữ liệu hình ảnh. Trong khi tác vụ phát hiện đối tượng (Object Detection) tập trung vào việc xác định vị trí của các vật thể bằng cách xây dựng các hộp giới hạn (bounding box), và phân đoạn ảnh (Image Segmentation) cung cấp thông tin chi tiết hơn về kích thước và hình dạng của các vật thể, thì phân loại ảnh tập trung vào việc trả lời câu hỏi: Hình ảnh này thuộc loại nào?



Hình 1: Image Classification, Object Detection and Instance Segmentation

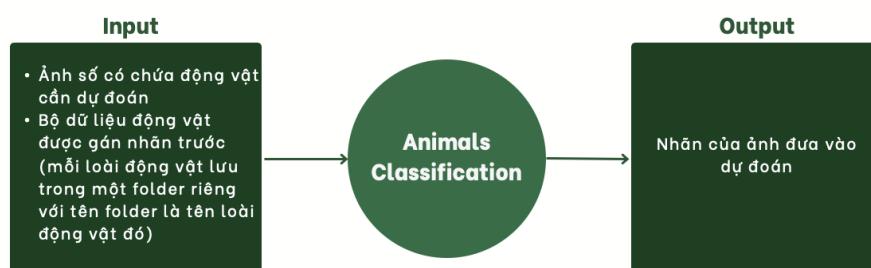
Phân loại ảnh tập trung vào nhiệm vụ tự động phân loại và gắn nhãn cho hình ảnh dựa trên dữ liệu đã được gán nhãn trước, đem lại nhiều ứng dụng thực tiễn. Phải kể đến phân loại ảnh chó mèo, đây là một trong những bài toán kinh điển trong thị giác máy tính và máy học. Từ bài toán cơ bản này, nhóm đã phát triển và mở rộng phạm vi nghiên cứu lên thành bài toán Phân loại động vật (Animals Classification). Việc phân loại ảnh các loài động vật trong tập dữ liệu cho trước, đặc biệt hữu ích trong các dự án nghiên cứu về động vật, giúp cho việc theo dõi và quản lý dữ liệu dễ dàng và hiệu quả hơn. Ngoài ra, có thể áp dụng trong các camera ở sở thú, giúp theo dõi và quản lý động vật một cách tự động và chính xác. Đó là lý do nhóm chúng em chọn đề tài Animals Classification.

1.2 Phát biểu bài toán

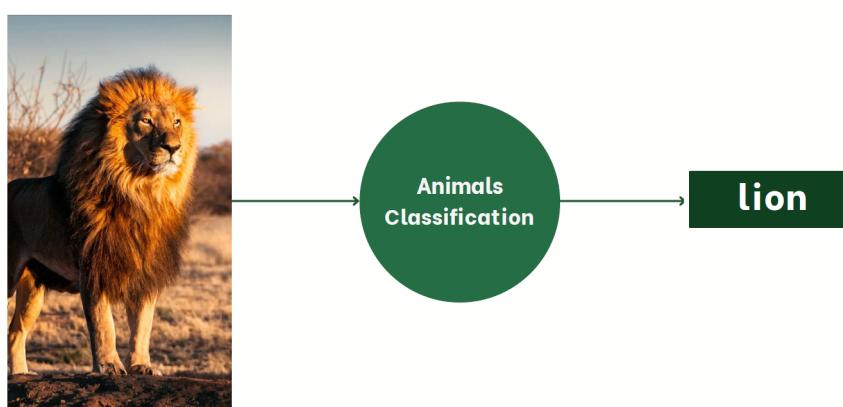
- Đề tài: Animals Classification

- **Input:** Ảnh số có chứa động vật cần dự đoán và bộ dữ liệu động vật được gán nhãn trước (mỗi loài động vật lưu trong một folder riêng với tên folder là tên loài động vật đó).

- **Output:** Nhãn của bức ảnh đưa vào dự đoán.



Hình 2: Input và Output của bài toán



Hình 3: Minh họa Input và Output bài toán

1.3 Nội dung thực hiện

Trong báo cáo này, nhóm sẽ thực hiện các nội dung sau:

- Rút trích đặc trưng của ảnh bằng HOG.
- Chuẩn hóa và giảm chiều dữ liệu bằng PCA.
- Sử dụng các phương pháp máy học SVM, KNN, RF để phân loại.
- Chạy thực nghiệm và đánh giá các phương pháp.

2 Cơ sở lý thuyết

2.1 Histogram Of Oriented Gradient - HOG

2.1.1 Giới thiệu HOG

Phương pháp HOG (Histogram of Oriented Gradients) là một kỹ thuật trích xuất đặc trưng được phát triển bởi Navneet Dalal và Bill Triggs giải quyết bài toán Human Detection vào năm 2005. Đây là một phương pháp phổ biến trong xử lý ảnh và thị giác máy tính, thường được sử dụng để nhận diện đối tượng và phân loại hình ảnh.

2.1.2 Các thuật ngữ

Một số thuật ngữ thường gặp trong HOG:

- **Feature Descriptor:** Bộ mô tả đặc trưng, là một phép biến đổi dữ liệu thành các đặc trưng giúp ích cho phân loại hoặc nhận diện vật thể. Các phương pháp có thể kể đến như HOG, SUFT, SHIFT.
- **Histogram:** Là biểu đồ histogram biểu diễn phân phối của các cường độ màu sắc theo khoảng giá trị.
- **Gradient:** Là đạo hàm của véc tơ cường độ màu sắc giúp phát hiện hướng di chuyển của các vật thể trong hình ảnh.
- **Local cell:** Ô cục bộ. Trong thuật toán HOG, một hình ảnh được chia thành nhiều cells bởi một lưới ô vuông. Mỗi cell được gọi là một ô cục bộ.
- **Local portion:** Vùng cục bộ hay còn gọi là block. Là một vùng trước trích suất ra từ ô vuông trên hình ảnh.
- **Local normalization:** Phép chuẩn hóa được thực hiện trên một vùng cục bộ. Thường là chia cho norm chuẩn bậc 2 hoặc norm chuẩn bậc 1. Mục đích của việc chuẩn hóa là để đồng nhất các giá trị cường độ màu sắc về chung một phân phối. Ta sẽ làm rõ hơn trong phần trình bày thuật toán.
- **Phương gradient** là độ lớn góc giữa vector gradient \vec{x} và \vec{y} giúp xác định phương thay đổi cường độ màu sắc, hay chính là phương đổ bóng của hình ảnh. Giả sử G_x và G_y lần lượt là giá trị gradient theo phương x và phương y của hình ảnh. Khi đó, phương gradient được tính như sau:

$$\theta = \arctan \left(\frac{G_y}{G_x} \right)$$

- Độ lớn gradient là chiều dài của vector gradient theo phương x và phương y . Biểu diễn phân phối histogram của vector này theo vector phương gradient sẽ thu được vector mô tả đặc trưng HOG. Độ lớn gradient được tính như sau:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

2.1.3 Nguyên lý hoạt động

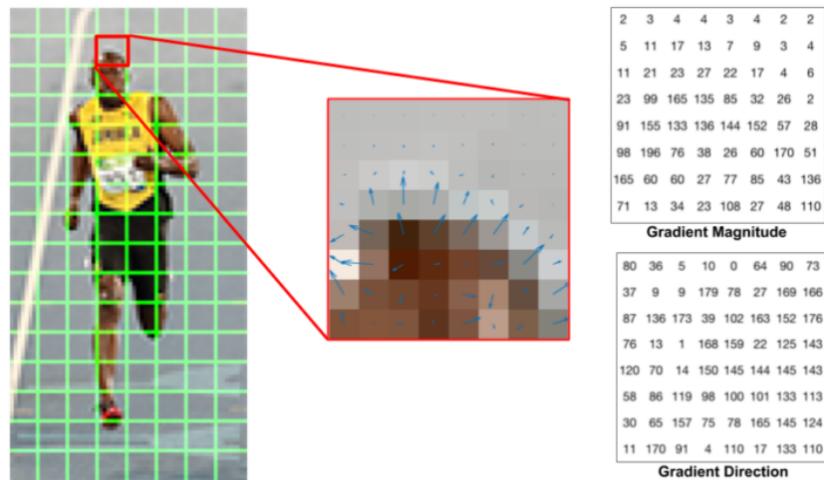
Nguyên lý hoạt động của HOG đó là hình dạng của một vật thể cục bộ có thể được mô tả thông qua hai ma trận đó là ma trận độ lớn gradient (**gradient magnitude**) và ma trận phương gradient (**gradient direction**). Vậy 2 ma trận gradient trên được tạo ra như thế nào? Đầu tiên hình ảnh được chia thành 1 lưới ô vuông và trên đó chúng ta xác định rất nhiều các vùng cục bộ liền kề hoặc chồng lấn lên nhau. Một vùng cục bộ bao gồm nhiều ô cục bộ (trong thuật toán HOG là 4) có kích thước là 8x8 pixels. Sau đó, một biểu đồ histogram thống kê độ lớn gradient được tính toán trên mỗi ô cục bộ. Bộ mô tả HOG (HOG descriptor) được tạo thành bằng cách nối liền (concatenate) 4 vec tơ histogram ứng với mỗi ô thành một vec tơ tổng hợp. Để cải thiện độ chính xác, mỗi giá trị của vec tơ histogram trên vùng cục bộ sẽ được chuẩn hóa theo norm chuẩn bậc 2 hoặc bậc 1. Phép chuẩn hóa này nhằm tạo ra sự bất biến tốt hơn đối với những thay đổi trong chiếu sáng và đổ bóng.

Các bước tính HOG:

- Bước 1: Tiền xử lý ảnh (đưa ảnh về một tỉ lệ nhất định).
- Bước 2: Tính gradient của ảnh.
- Bước 3: Tính Histogram of Gradients trong từng cell (ô cục bộ).
- Bước 4: Chuẩn hóa Blocks (vùng cục bộ).
- Bước 5: Tính HOG feature vector.

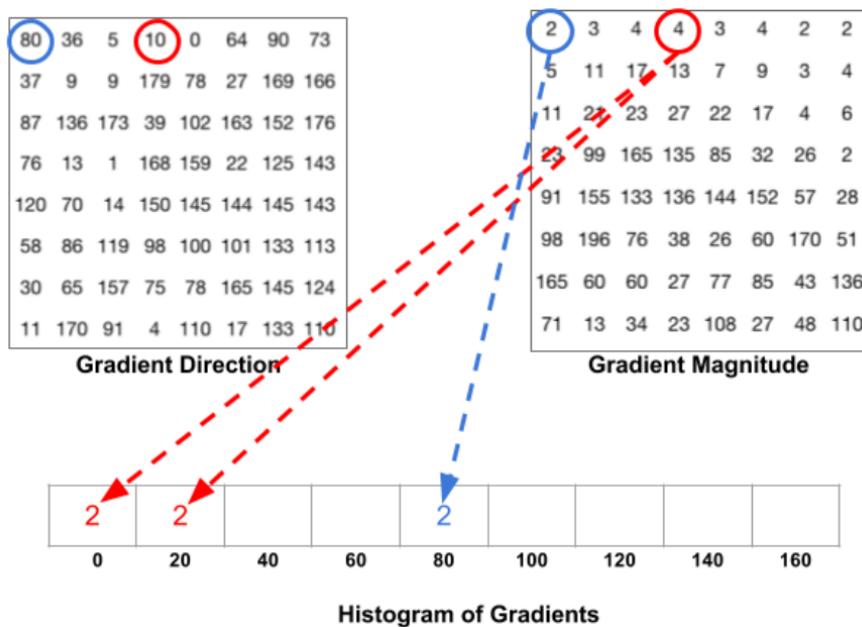
Áp dụng trong thực nghiệm:

- Đầu tiên các bức ảnh gốc được resize lại thành (64x64) và chuyển thành ảnh xám.
- Tiếp theo chia hình ảnh thành một lưới ô vuông mà mỗi một ô có kích thước 8x8 pixels. Như vậy có tổng cộng 64 ô pixels tương ứng với mỗi ô.
- Trên mỗi cell ta sẽ cần tính ra 2 tham số đó là độ lớn gradient (gradient magnitude) và phương gradient (gradient direction).



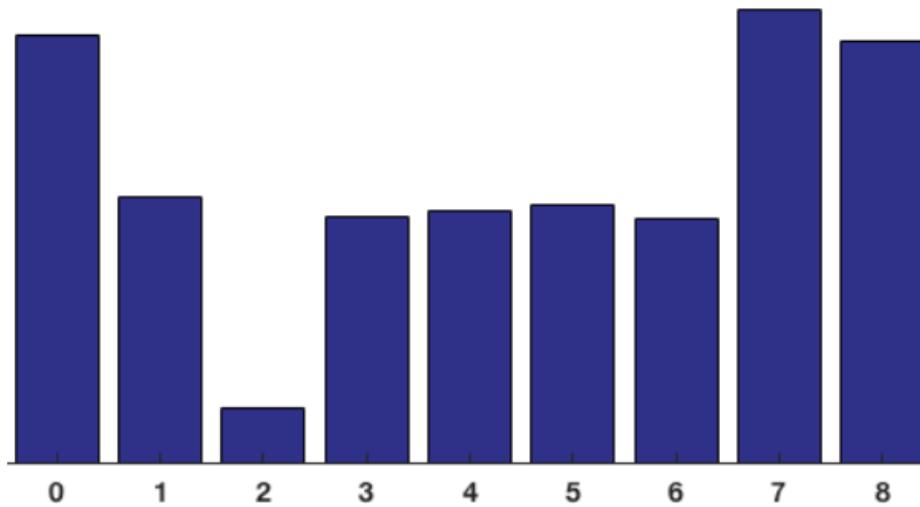
Hình 4: Chia cells và tính gradient magnitude + gradient direction

- Mapping độ lớn gradient vào các bins tương ứng của phương gradient.



Hình 5: Mapping độ lớn gradients với các bins.

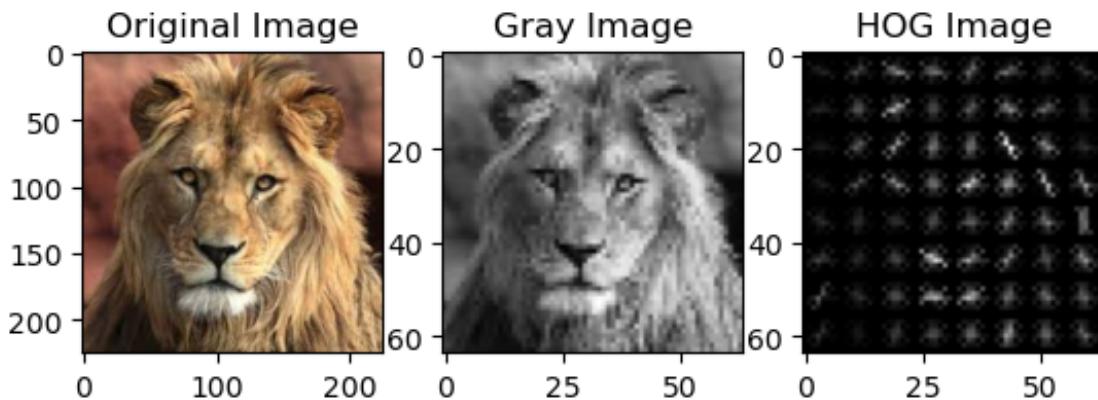
- Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của véc tơ bins ta thu được biểu đồ Histogram of Gradients
- Chuẩn hóa véc tơ histogram theo block 2x2. Với mỗi block 2x2 sẽ có 4 vector histogram 1x9, concatenate các véc tơ sẽ thu được véc tơ histogram tổng hợp kích thước là 1x36 và sau đó chuẩn hóa theo norm chuẩn bậc 2 trên véc tơ này. Thực hiện di chuyển



Hình 6: Biểu đồ Histogram of Gradient gồm 9 bins (1 cell)

các window tương tự như phép tích chập 2 chiều trong mạng CNN với step_size = 8 pixels.

- Sau khi chuẩn hóa các véc tơ histogram, chúng ta sẽ concatenate các véc tơ 1×36 này thành một véc tơ lớn. Đây chính là véc tơ HOG đại diện cho toàn bộ hình ảnh.



Hình 7: Ví dụ về HOG

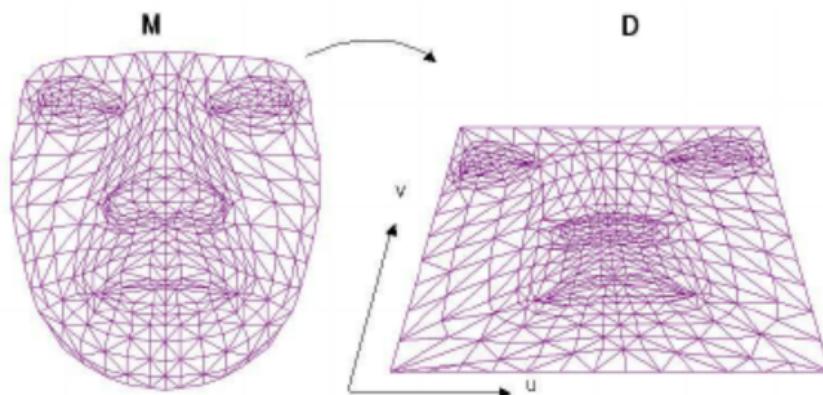
Các tham số sử dụng trong HOG:

- orientations = [9, 18, 36]. Cụ thể với orientations=9 số chiều của feature là **1764**, orientations=18 số chiều của feature là **3528** và orientations=36 số chiều của feature là **7056**.
- Các tham số còn lại sử dụng tham số mặc định.

2.2 Principal Component Analysis - PCA

Phương pháp Phân tích thành phần chính (PCA) không phải là công trình của một người duy nhất mà là sự kết hợp của nhiều công trình và nghiên cứu. Pearson đã đưa ra ý tưởng ban đầu (1901), và sau đó Hotelling (1933) đã có những đóng góp quan trọng cho việc phát triển và phổ biến PCA như chúng ta đã biết ngày nay.

Thuật toán PCA (Principal Component Analysis) là một trong những phương pháp giảm chiều dữ liệu trong khi vẫn giữ lại hầu hết các đặc trưng quan trọng của nó. PCA được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau, từ xử lý ảnh đến phân tích dữ liệu y tế và tài chính. Việc giảm chiều dữ liệu là một bước quan trọng trong việc xử lý dữ liệu lớn và phức tạp, giúp giảm chi phí tính toán, tăng tốc độ xử lý và cải thiện độ chính xác của các mô hình dự đoán.



Hình 8: Giảm chiều dữ liệu từ 3D xuống 2D

Các tham số sử dụng trong PCA:

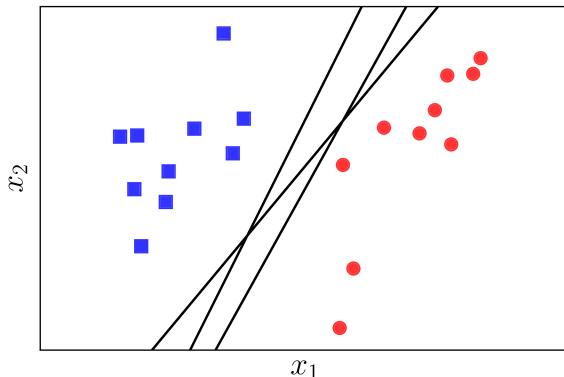
- `n_components` = [None, 50, 100, 200, 500]
- Các tham số còn lại sử dụng tham số mặc định.

2.3 Support Vector Machine - SVM

Support Vector Machine được phát triển vào những năm 1990 bởi Vladimir N. Vapnik và các đồng nghiệp của ông, và họ đã xuất bản công trình này trong một bài báo có tựa đề "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing" năm 1995.

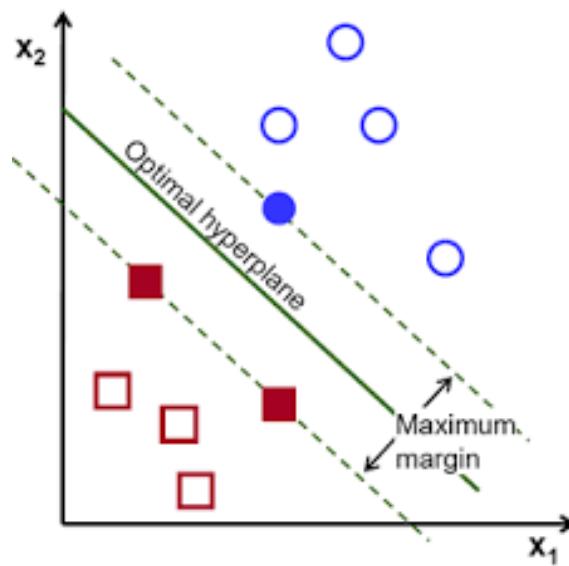
SVM có thể được sử dụng trong cả phân loại và hồi quy. Tuy nhiên, nó chủ yếu được sử dụng để phân loại. Nếu được sử dụng trong bài toán phân loại, Support Vector Machine được gọi là Support Vector

Classifier (SVC). Trong SVM, mục tiêu của thuật toán là tìm một siêu phẳng (hyperplane) trong không gian N chiều (N là số đặc trưng của tập dữ liệu) để phân tách dữ liệu thành các lớp tương ứng. Câu hỏi đặt ra là: Có rất nhiều siêu mặt phẳng có thể phân tách dữ liệu thành các lớp tương ứng, vậy thì nên chọn mặt phẳng nào?



Hình 9: Các siêu mặt phẳng (hyperplane)

Mục tiêu của SVM là tìm một siêu phẳng có biên (margin) lớn nhất, tức là khoảng cách giữa siêu phẳng và các điểm dữ liệu gần siêu phẳng nhất (hay còn gọi là support vectors). Biên rộng hơn tạo ra sự phân chia tốt hơn giữa các lớp, cải thiện độ tin cậy của mô hình.

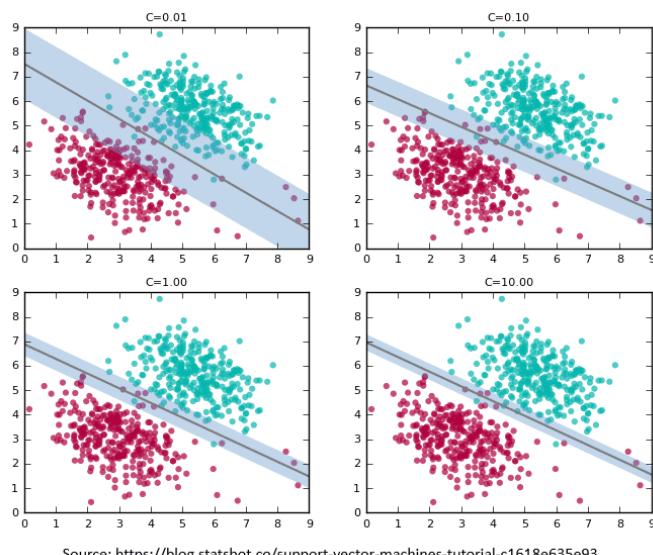


Hình 10: Siêu phẳng có margin lớn nhất của 2 lớp

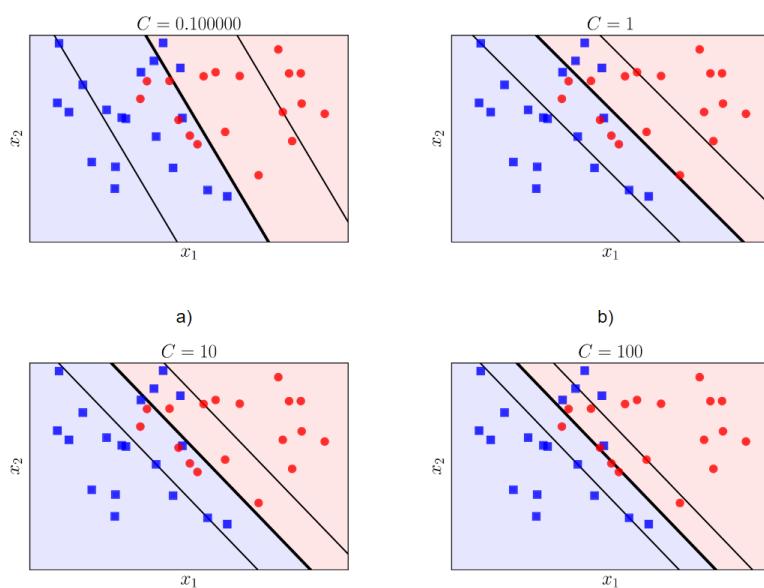
Một biến thể khác của SVM là **Soft Margin SVM**. Đối với dữ không phân biệt tuyến tính (non linear separable), chúng ta không thể vẽ được đường thẳng hay mặt phẳng để phân loại các điểm dữ liệu thành 2 lớp riêng biệt. Thuật toán này cho phép SVM mắc một số lỗi nhất

định với mục tiêu giữ cho lề càng rộng càng tốt và các điểm khác vẫn được phân loại chính xác. Nói cách khác, nó sẽ cân bằng giữa việc phân loại sai và tối đa hóa lề. Sự cho phép vi phạm một số điểm dữ liệu giúp giảm thiểu overfitting (quá khớp) và làm cho mô hình SVM linh hoạt hơn trong việc xử lý các tập dữ liệu có độ phức tạp cao hơn.

Một đại lượng quan trọng và để kiểm soát mức độ sai sót của Soft Margin SVM là mức độ chấp nhận lỗi (hay tham số C). Khi tham số C càng lớn nghĩa là SVM bị phạt càng nặng khi phân loại sai và kết quả sẽ cho ra 1 lề (margin) hẹp và ngược lại.



Hình 11: Tham số C trong SVM

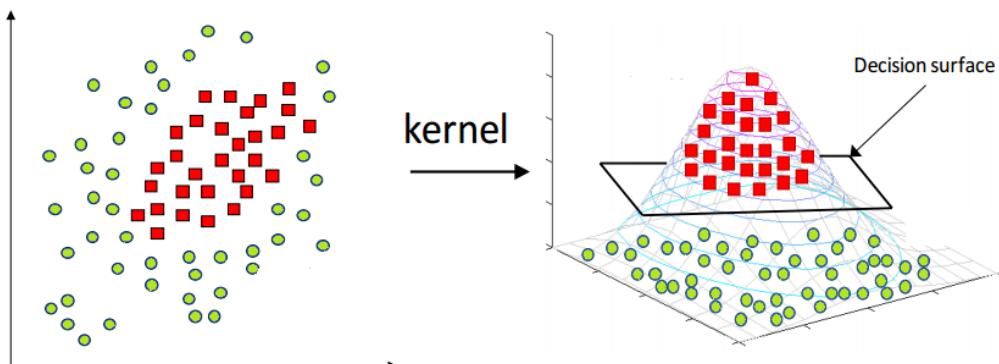


Hình 12: Ảnh hưởng của C lên nghiệm

Kernel SVM (Support Vector Machine) là một phiên bản mở rộng của SVM cho phép xử lý các tập dữ liệu không tuyến tính bằng cách sử dụng hàm nhân (kernel function). Thay vì tìm một siêu phẳng trong không gian ban đầu, Kernel SVM ánh xạ dữ liệu vào không gian nhiều chiều hơn thông qua hàm kernel và sau đó tìm một siêu phẳng phân chia tuyến tính trong không gian mới. Các hàm nhân phổ biến được sử dụng bao gồm hàm đa thức (polynomial kernel), hàm Radial Basis Function (RBF kernel), và hàm sigmoid.

Tên	Công thức	kernel
linear	$\mathbf{x}^T \mathbf{z}$	'linear'
polynomial	$(r + \gamma \mathbf{x}^T \mathbf{z})^d$	'poly'
sigmoid	$\tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$	'sigmoid'
rbf	$\exp(-\gamma \ \mathbf{x} - \mathbf{z}\ _2^2)$	'rbf'

Hình 13: Các loại kernel



Hình 14: Dữ liệu ban đầu và dữ liệu sau khi áp dụng kernel

Các tham số sử dụng trong SVM:

- Kernel: 'linear', 'poly', 'rbf', 'sigmoid'.
- C: 0.01, 0.1, 1, 10.

2.4 k-Nearest Neighbors - KNN

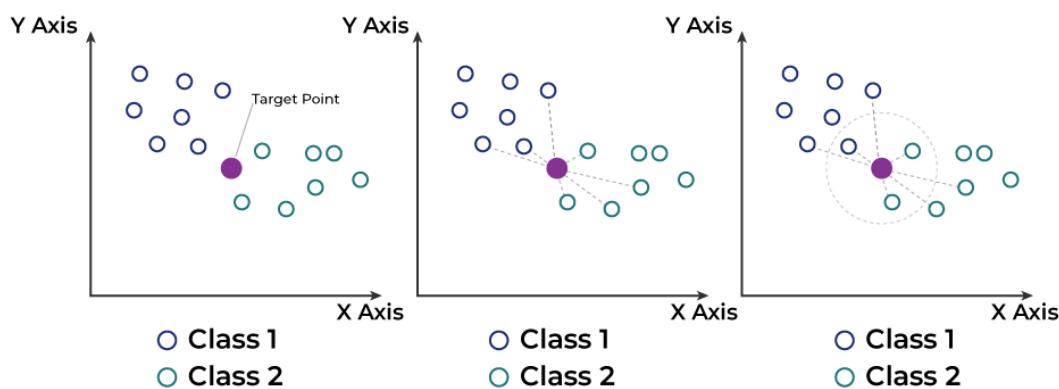
Thuật toán k-Nearest Neighbors hay gọi tắt là KNN, được **Evelyn Fix** và **Joseph Hodges** công bố đầu tiên vào năm 1951 và được **Thomas Cover** mở rộng sau này vào năm 1967.

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression.

Với KNN trong bài toán Classification, label của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.

Các bước thực hiện phân loại bằng KNN:

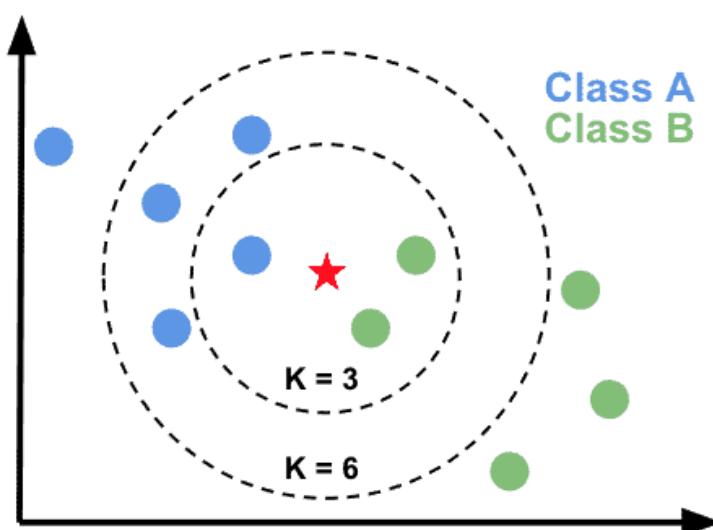
- Bước 1: Có **D** là tập các điểm dữ liệu đã được gán nhãn, **A** là dữ liệu chưa được phân loại. Chọn tham số **k** tùy ý.
- Bước 2: Đo khoảng cách từ dữ liệu mới **A** đến tất cả các dữ liệu khác đã được phân loại trong **D**.
- Bước 3: Chọn **K** điểm dữ liệu trong **D** gần với **A** nhất.
- Bước 4: Kiểm tra danh sách **k** điểm đã chọn và đếm số lượng của mỗi lớp xuất hiện.
- Bước 5: Trả về nhãn của **A** là lớp xuất hiện nhiều nhất trong **k** điểm dữ liệu phía trên



Hình 15: Minh họa thuật toán KNN

Các tham số sử dụng trong KNN:

- **n_neighbors**: Đây là siêu tham số quan trọng nhất trong KNN, là số n hàng xóm gần nhất quyết định nhãn của dữ liệu đầu vào. Nhóm thực nghiệm với n trong khoảng [1,29] để khảo sát sự biến thiên và chọn k phù hợp nhất.
- **weight**: Trọng số của các điểm hàng xóm lân cận. Nhóm sử dụng tham số mặc định là '**uniform**'.



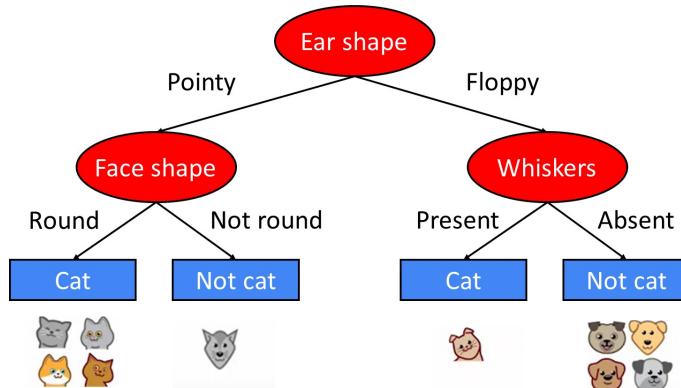
Hình 16: Minh họa tầm quan trọng của tham số n_neighbors (k)

2.5 Random Forest - RF

Thuật toán Random Forest được giới thiệu bởi **Leo Breiman** và **Adele Cutler** vào năm 2001. Random Forest là thuật toán supervised learning, có thể giải quyết cả bài toán regression và classification.

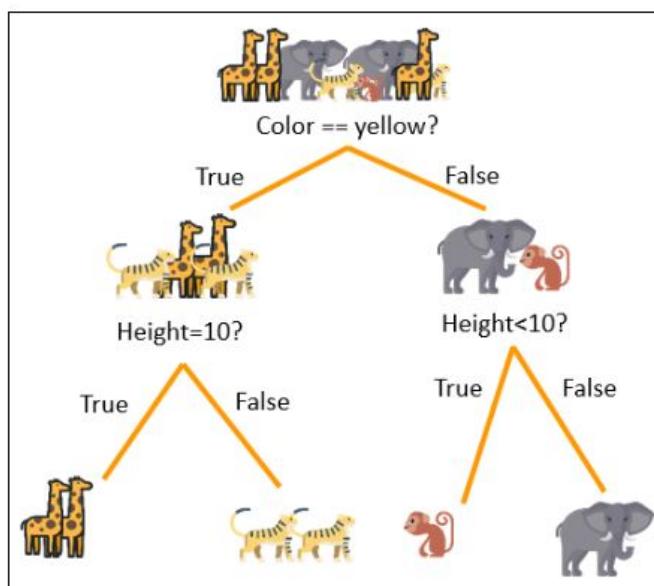
Random là ngẫu nhiên, Forest là rừng, nên Random Forest có ý tưởng là xây dựng nhiều cây quyết định, tuy nhiên mỗi cây sẽ khác nhau (có yếu tố random). Kết quả dự đoán cuối cùng được tổng hợp từ các cây quyết định.

Cây quyết định (decision tree) bao gồm một tập hợp các nút, trong đó mỗi nút đại diện cho một quyết định hoặc một dự đoán. Cây bắt đầu từ nút gốc và kết thúc ở các nút lá, mỗi nút lá đại diện cho một dự đoán hoặc một lớp/phân loại cuối cùng. Cây được xây dựng dựa trên bộ dữ liệu huấn luyện, bằng cách chọn ra đặc trưng tốt nhất để chia dữ liệu thành các nhóm.



Hình 17: Ví dụ về decision tree phân loại chó mèo

Khi có một dữ liệu mới cần dự đoán, Random Forest sẽ trả về dự đoán cuối cùng bằng cách kết hợp dự đoán của tất cả các cây quyết định. Đối với bài toán phân loại, kết quả cuối cùng thường được quyết định bằng cách bầu chọn (voting) giữa các cây quyết định; và đối với bài toán hồi quy, kết quả cuối cùng thường là trung bình của các dự đoán từ các cây quyết định.



Hình 18: Minh họa sử dụng decision tree để phân loại động vật

Các bước thực hiện phân loại bằng Random Forest:

- Bước 1: Từ tập dữ liệu huấn luyện ban đầu, tạo n tập con dữ liệu ngẫu nhiên. Chọn tham số **n** tùy ý.
- Bước 2: Xây dựng n cây quyết định dựa trên n tập con dữ liệu bên trên.

- Bước 3: Khi có dữ liệu mới cần phân loại, RF sẽ sử dụng tất cả cây quyết định đã được huấn luyện để tổng hợp và đưa ra dự đoán cuối cùng.

Các tham số sử dụng trong Random Forest:

- **n_estimators:** Đây là tham số quan trọng nhất trong RF, là số cây quyết định sẽ được tạo ra.

3 Thực nghiệm

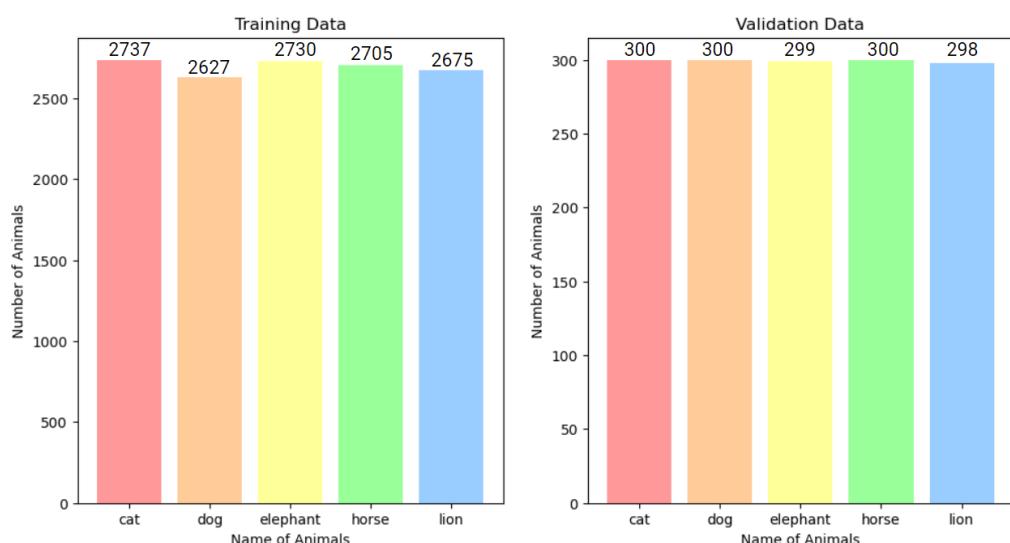
3.1 Dataset

- Source Kaggle: Animals Dataset



Hình 19: 5 loài động vật trong dataset

- Trong dataset gốc trên Kaggle, dữ liệu được chia làm 2 folder Train (13473) với Val (1497), mỗi folder có 5 folders con là dữ liệu của 5 loài động vật. Số lượng dữ liệu mỗi loài trong mỗi tập được thống kê trong biểu đồ dưới đây:



Hình 20: Thống kê dữ liệu

- Từ biểu đồ thống kê ta thấy, số lượng dữ liệu của mỗi loài động vật không chênh lệch nhiều và khá cân bằng.
- Để tiến hành thực nghiệm, nhóm sử dụng tập val gốc để làm tập test, sau đó tiếp tục chia tập train gốc thành 2 tập là train và validation với tỷ lệ tương ứng là 7:3. Sau khi chia dữ liệu ta được:
 - + **Train - 9432**: huấn luận mô hình học máy.
 - + **Val - 4042**: đánh giá hiệu suất của mô hình và điều chỉnh các tham số của mô hình
 - + **Test - 1497**: đánh giá hiệu suất cuối cùng của mô hình sau khi đã hoàn tất quá trình huấn luyện và điều chỉnh tham số.

3.2 Metrics

- **Accuracy** là một phép đo đánh giá hiệu suất của một mô hình dự đoán trong machine learning. Nó đo lường tỷ lệ các dự đoán chính xác so với tổng số lượng các dự đoán.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total sample}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision** trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng ?

$$\text{Precision} = \frac{TP}{\text{Total predicted positive}} = \frac{TP}{TP + FP}$$

- **Recall** đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm positive. Công thức của recall như sau:

$$\text{Recall} = \frac{TP}{\text{Total actual positive}} = \frac{TP}{TP + FN}$$

- **F1 Score** là trung bình điều hòa giữa precision và recall.

$$\text{F1 Score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

3.3 Cài đặt

Các tham số chạy thực nghiệm:

- **HOG**: orientations = [9, 18, 36]
- **PCA**: n_components = [None, 50, 100, 200, 500]
- **SVM**: Kernel = ['linear', 'poly', 'rbf', 'sigmoid'] và C = [0.01, 0.1, 1, 10]
- **KNN**: k = (0, 30)
- **RF**: default n_estimators = 100

⇒ Với mỗi thuật toán máy học chạy hết tất cả các tổ hợp tham số

3.4 Đánh giá

3.4.1 Kết quả thực nghiệm trên tập validation

- **HOG và PCA**: với mỗi thuật toán máy học chạy hết các tổ hợp tham số. Sau đó tổng hợp và tính **accuracy trung bình** theo tham số orientations (HOG) và n_components (PCA) để chọn ra bộ tham số tốt nhất. Dựa vào kết quả thực nghiệm, ta thấy với **orientations=36** và **n_components=50** thì accuracy trung bình đạt được cao nhất. Nhóm sẽ sử dụng bộ tham số này để chạy thực nghiệm so sánh các mô hình máy học SVM, KNN và RF.

orientations \ components	50	100	200	500	NoPCA
9	0.6955	0.688	0.6716	0.6506	0.6445
18	0.706	0.6935	0.6769	0.6566	0.6424
36	0.7064	0.6903	0.6715	0.6616	0.6321

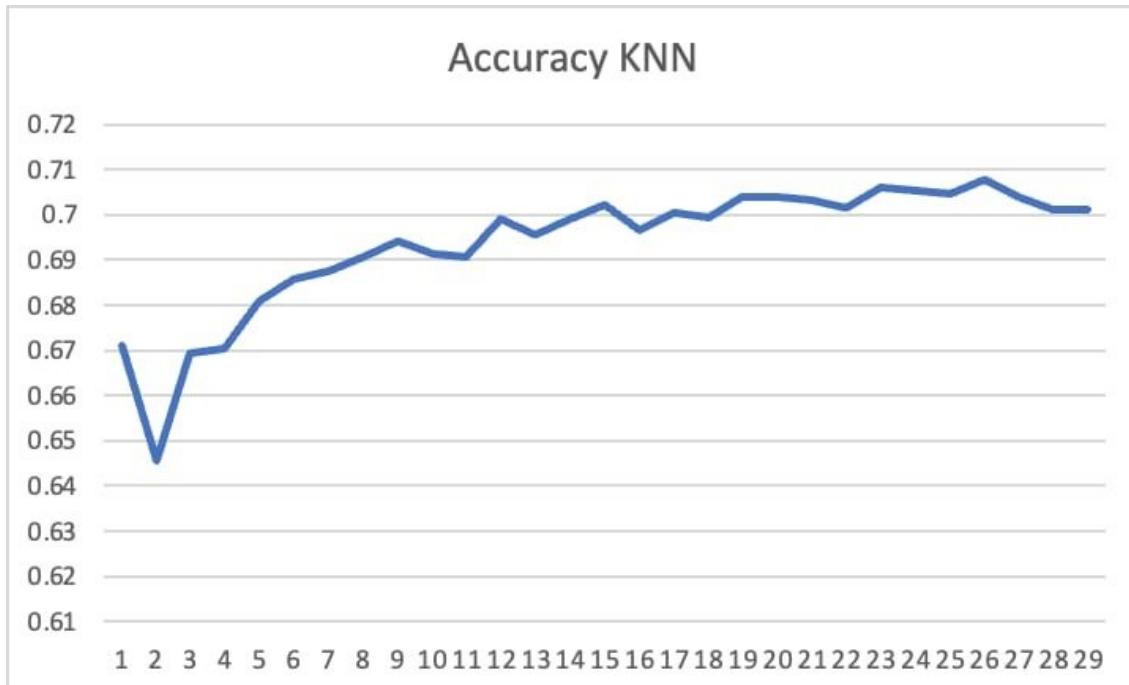
Hình 21: Accuracy TB của tổ hợp orientations và n_components

- **SVM:** chạy thuật toán SVM với đặc trưng sử dụng orientations=36 và n_components=50 để chọn ra bộ tham số tốt nhất cho SVM. Dựa vào kết quả thực nghiệm, với tham số **C=1** và **kernel=rbf** thì accuracy cao nhất.

C \ kernel	linear	poly (degree=3)	rbf	sigmoid
C=0.01	0.7142	0.4426	0.6037	0.6848
C=0.1	0.7115	0.6710	0.7341	0.7053
C=1	0.7119	0.7573	0.7787	0.6456
C=10	0.7119	0.7516	0.7720	0.6060

Hình 22: Accuracy của tổ hợp tham số kernel và C trong SVM

- **KNN:** chạy thuật toán KNN với đặc trưng sử dụng orientations=36 và n_components=50 để chọn ra tham số tốt nhất cho KNN. Dựa vào kết quả thực nghiệm, với tham số **k=26** thì accuracy cao nhất.



Hình 23: Accuracy KNN theo tham số k

- So sánh các thuật toán máy học:

	SVM (kernel='rbf',C=1)	KNN (k=26)	RF (n_estimators = 100)
Accuracy	77.12%	70.78%	71.33%
Precision	76.99%	70.77%	71.11%
Recall	77.12%	70.78%	71.33%
F1	77.03%	69.98%	71.09%

Hình 24: Kết quả thực nghiệm của SVM, KNN và RF trên tập Validation

3.4.2 Kết quả thực nghiệm trên tập Test

	SVM (kernel='rbf',C=1)	KNN (k=26)	RF (n_estimators = 100)
Accuracy	80.09%	72.75%	73.48%
Precision	80.02%	73.41%	73.14%
Recall	80.09%	72.75%	73.48%
F1	79.96%	72.06%	73.07%

Hình 25: Kết quả thực nghiệm của SVM, KNN và RF trên tập Test

- Trong 3 thuật toán phân loại thì SVM cho độ chính xác vượt trội nhất, KNN và RF có độ chính thấp hơn và xác xấp xỉ nhau.
- SVM có thời gian tính toán để tìm được margin tối ưu lâu hơn so với KNN và RF.
- Random Forest dễ bị overfitting khi số lượng n_estimators quá lớn.

4 Nhận xét và kết luận

4.1 Ưu điểm

- Rút trích đặc trưng bằng HOG hiệu quả với các hình ảnh có cấu trúc và biên rõ ràng như các loài động vật. HOG đơn giản trong việc tính toán nhưng vẫn giữ được độ chính xác tương đối cao. Đồng thời PCA giúp giảm chiều dữ liệu, loại bỏ nhiễu và

tối ưu hóa việc lưu trữ và xử lý dữ liệu, giúp tăng tốc độ huấn luyện và dự đoán của các mô hình máy học.

- Chỉ sử dụng phương pháp rút trích đặc trưng cơ bản và các mô hình máy học đơn giản đã cho độ chính xác trong khoảng từ 70% đến 80%.

4.2 Hạn chế

- **Dữ liệu:** chưa được đồng nhất. Với chó và mèo thì hình ảnh được chụp cận mặt, còn voi, ngựa và sư tử thì được chụp toàn thân và chứa nhiều background hơn.
- **Độ chính xác:** với cách tiếp cận đơn giản thì đề tài chỉ dừng lại ở việc phân loại một số ít động vật, khi số lượng lớp động vật tăng lên thì hiệu quả của mô hình càng kém đi.

4.3 Hướng phát triển

- **Cải thiện thuật toán rút trích đặc trưng:** nghiên cứu và ứng dụng các kỹ thuật rút trích đặc trưng hiện đại hơn như Convolutional Neural Networks (CNN), Deep Learning để thay thế cho HOG và PCA.
- **Tối ưu hóa tham số:** sử dụng các phương pháp tối ưu hóa tự động như Grid Search hoặc Random Search kết hợp với Cross-Validation để tìm ra bộ tham số tốt nhất cho các mô hình.
- **Tăng cường dữ liệu:** thu thập thêm dữ liệu để mở rộng khả năng phân loại của mô hình.

5 Tài liệu tham khảo

[1] <https://phamdinhkhanh.github.io/2019/11/22/HOG.html>

[2] <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

[3] <https://machinelearningcoban.com/2017/06/15/pca/>

[4] <https://www.geeksforgeeks.org/k-nearest-neighbours/>

[5] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).
<https://doi.org/10.1023/A:1010933404324>