ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

NGUYỄN TRỌNG ÂN
DƯƠNG VĂN BÌNH

ĐỒ ÁN CUỐI KỲ

# XÂY DỰNG MÔ HÌNH NHẬN DIỆN HÌNH ẢNH TRONG NHẬN DIỆN HÌNH ẢNH TRÁI CÂY ĐƯỢC CHỤP TỪ NHIỀU GÓC ĐỘ

# BUILDING A MODEL FOR DETECTING FRUITS FROM IMAGES TAKEN FROM DIFFERENT ANGLES

CỬ NHÂN NGÀNH KHOA HỌC DỮ LIỆU

TP. HỒ CHÍ MINH - 2020

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

NGUYỄN TRỌNG ÂN – 18520434

DƯƠNG VĂN BÌNH – 18520505

ĐỒ ÁN CUỐI KỲ

# XÂY DỰNG MÔ HÌNH NHẬN DIỆN HÌNH ẢNH TRONG NHẬN DIỆN HÌNH ẢNH TRÁI CÂY ĐƯỢC CHỤP TỪ NHIỀU GÓC ĐỘ

# BUILDING A MODEL FOR DETECTING FRUITS FROM IMAGES TAKEN FROM DIFFERENT ANGLES

CỬ NHÂN NGÀNH KHOA HỌC DỮ LIỆU

GIẢNG VIÊN HƯỚNG DẪN

TS. NGUYỄN TẤN TRẦN MINH KHANG

Th.S. VÕ DUY NGUYÊN

TP. HỒ CHÍ MINH - 2020

# THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

# LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn tới thầy Nguyễn Tấn Trần Minh Khang và thầy Võ Duy Nguyên trong suốt thời gian môn học đã truyền đạt những kiến thức tốt nhất và phổ biến nhất có thể trong môn Học máy thống kê (DS102.K21).

Cảm ơn thầy Võ Duy Nguyên đã giúp đỡ chúng em để có thể khắc phục những khó khăn trong việc sử dụng dữ liệu để xây dựng mô hình trong đồ án này.

MỤC LỤC

# DANH MỤC HÌNH

# DANH MỤC BẢNG

# DANH MỤC TỪ VIẾT TẮT

**KNN: K-NEAREST NEIGHBORS**

**SVM: SUPPORT VECTOR MACHINE**

**CNNs: CONVOLUTIONAL NEURAL NETWORKS**

**MIT: Massachusetts Institute of Technology**

**HOG: Histogram of Oriented Gradients**

# TÓM TẮT ĐỒ ÁN

Đồ án xây dựng mô hình nhận diện trái cây và rau từ bộ dữ liệu Fruits-360, chứa hình ảnh được chụp từ nhiều góc độ.

Trong đồ án sử dụng các mô hình K-nearest neighbors, Support Vector Machine, Convolutional Neural Networks.

Với mô hình KNN sử dụng các độ đo khoảng cách Minkowski, Euclidean, với các tham số neighbors là 3, 5, 7 để tìm được tham số mô hình thích hợp cho mô hình KNN.

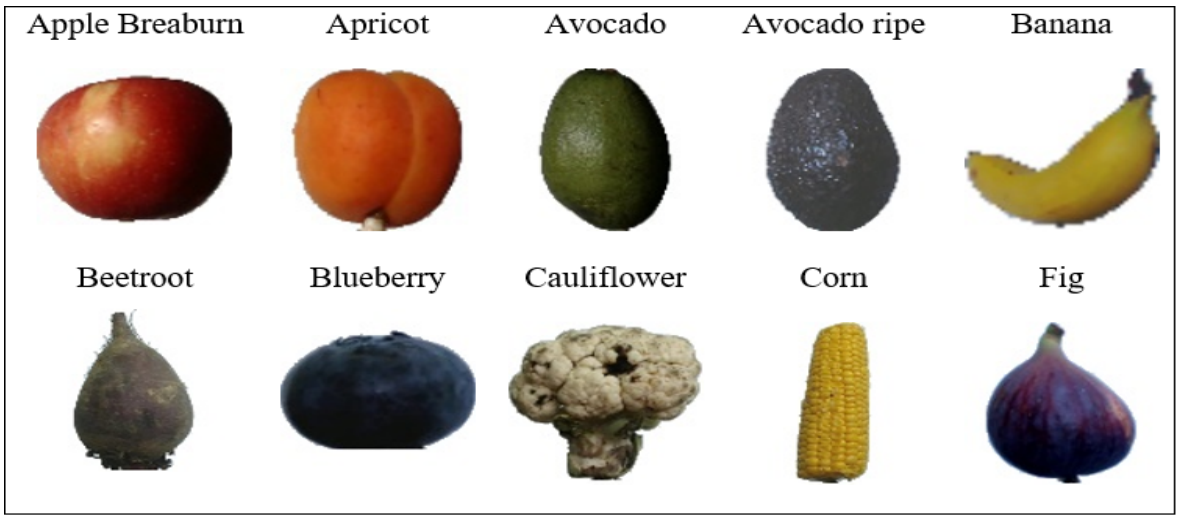
# Chương 1. **TỔNG QUAN**

## 1.1. Mục tiêu đề tài

Mục tiêu của đề tài là áp dụng các kiến thức đã được trang bị trong môn Học máy thống kê để xây dựng một mô hình nhận diện hình ảnh (trái cây và rau), qua bộ dữ liệu có sẵn Fruits-360.

Trong phạm vi của bộ dữ liệu Fruits-360 có thể huấn luyện mô hình nhận diện 131 loại trái cây và rau khác nhau. Tuy nhiên, vì những khó khăn trong việc đáp ứng tài nguyên huấn luyện mô hình và để đảm bảo chất lượng mô hình, do đó trong phạm vi khóa luận này chỉ huấn luyện mô hình nhận dạng hình ảnh của 10 loại trái cây và rau: táo Breaburn (Apple Breaburn), mơ (Apricot), bơ (Avocado), bơ chín (Avocado ripe), chuối (Banana), củ rền (Beetroot), việt quất (Beetroot), súp lơ (Cauliflower), bắp (Corn), quả sung (Fig).

Việc xây dựng mô hình với 10 nhãn (10 loại trái cây và rau) trong khóa luận này sẽ là tài liệu tham khỏa cho những sinh viên trong việc phát triển các mô hình nhận diện trong tương lai giúp nhận diện các loại thực phẩm trong các siêu thị, nhà máy chế biến… . Nhằm giúp con người phân loại, đánh giá các sản phẩm một cách tự động.




Hình 1.1: Hình ảnh 10 nhãn trong dùng trong việc huấn luyện mô hình.

## 1.2. Giới thiệu về bộ dữ liệu Fruits-360

Bộ dữ liệu này được xây dựng bởi nhóm tác giả Mihai Oltean, Horea Muresan thuộc Viện công nghệ Massachusetts (MIT).

Thông tin về bộ dữ liệu:

- Tổng số hình ảnh: 90483 tấm ảnh.
- Kích thước tập huấn luyện: 67692 tấm ảnh (có duy nhất một loại quả hay rau trong mỗi hình).
- Kích thước tập kiểm thử: 22688 tấm ảnh (có duy nhất một loại quả hay rau trong mỗi hình).
- Kích thước tập Multi-fruits: 103 tấm ảnh (có nhiều hơn một loại quả hay rau trong mỗi hình).
- Số lượng nhãn: 131 (trái cây và rau).
- Kích thước ảnh: 100x100 pixels.
- Định dạng tệp tin: <chỉ số>_100.jpg (ví dụ: 32_100.jpg) hay r_<chỉ số>_100.jpg hay r2_<chỉ số>_100.jpg hay r3_<chỉ số>_100.jpg. “r” là viết tắt cho rotated fruits (trái cây bị xoay một góc nào đó). “r2” là trái cây bị xoay quanh trục z. “100” là kích thước của tấm hình (100x100 pixels).
- Với những trái cây có nhiều loại khác nhau (ví dụ như táo) thì mỗi loại sẽ là một nhãn riêng.

Hình 1.2: Hình ảnh nhãn 2 loại táo khác nhau Apple Red 1 (trái) và Apple Golden 2 (phải).

## 1.3. Cấu trúc đồ án

Chương 1. TỔNG QUAN

1.1. Mục tiêu đề tài
1.2. Giới thiệu về bộ dữ liệu Fruits-360
1.3. Cấu trúc khóa luận

Chương 2. CƠ SỞ LÝ THUYẾT

2.1. Thuật toán K-nearest neighbors (KNN)
2.2. Thuật toán Logistic Regression
2.3. Thuật toán Support Vector Machine (SVM)
2.4. Convolutional Neural Networks (CNNs)
2.5. Phương pháp mô tả đặc trưng HOG

Chương 3. QUÁ TRÌNH THỰC HIỆN

3.1. Tiền xử lý dữ liệu
3.2. Cài đặt các mô hình

Chương 4. KẾT QUẢ

Chương 5. KẾT LUẬN

# Chương 2. CƠ SỞ LÝ THUYẾT

## 2.1. Thuật toán K-nearest neighbors (KNN)[1]

### 2.1.1. Giới thiệu thuật toán

KNN là một trong các thuật toán phân loại đơn giản nhất.

KNN là một trong những phương pháp học máy có giám sát 'Supervised Learning' tức là dựa trên biến mục tiêu đã được xác định trước đó, thuật toán sẽ xem xét dữ liệu đã chứa biến mục tiêu (đã được phân loại) để 'học' và tìm ra những biến đặc trưng có thể tác động đến biến mục tiêu.

KNN dựa trên giả định là những thứ tương tự hay có tính chất gần giống nhau sẽ nằm ở vị trí gần nhau, với giả định như vậy, KNN được xây dựng trên các công thức toán học phục vụ để tính khoảng các giữa 2 điểm dữ liệu (data point) để xem xét mức độ giống nhau của chúng.

KNN là một thuật toán đơn giản, quá trình huấn luyện (train) không quá phức tạp để hoàn thiện mô hình. Điều này làm cho việc xây dựng mô hình nhanh hơn nhưng giai đoạn thử nghiệm chậm hơn và tốn kém hơn về mặt thời gian và bộ nhớ lưu trữ, đặc biệt khi bộ dữ liệu lớn và phức tạp với nhiều biến khác nhau. Trong trường hợp xấu nhất, KNN cần thêm thời gian để quét tất cả các điểm dữ liệu và việc này sẽ cần nhiều không gian bộ nhớ hơn để lưu trữ dữ liệu.

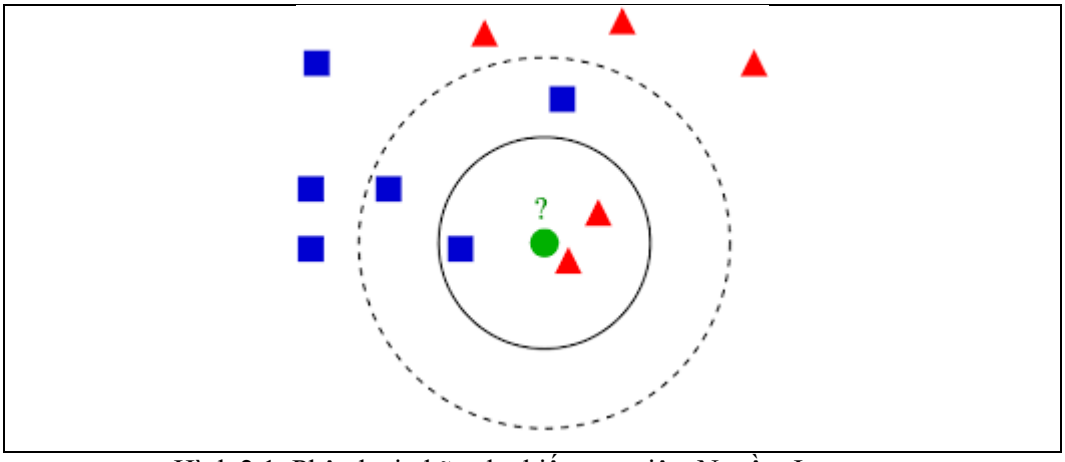
KNN không cần dựa trên các tham số khác nhau để tiến hành phân loại dữ liệu, không đưa ra bất kỳ kết luận cụ thể nào giữa biến đầu vào (input) và biến mục tiêu (target), mà chỉ dựa trên khoảng cách giữa điểm dữ liệu cần phân loại với điểm dữ liệu đã phân loại trước đó. Đây là một dặc điểm cực kỳ hữu ích vì hầu hết dữ liệu trong thế giới thực tại không thực sự tuân theo bất kỳ giả định lý thuyết nào.

### 2.1.2. Mô tả cách thức vận hành thuật toán

**Khoảng cách trong không gian vector:** Trong không gian một chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó. Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau, trong đó độ dài đường thẳng nối hai điểm chỉ là một trường hợp đặc biệt trong đó.

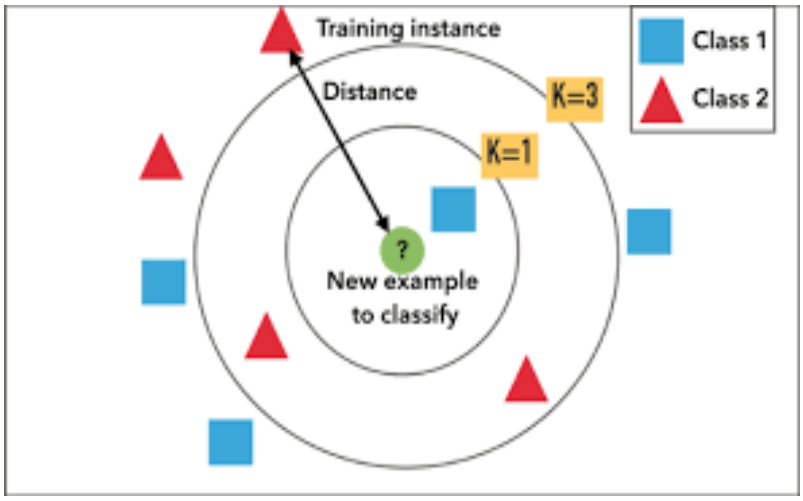
**Các bước thực hiện thuật toán KNN:**

- **Bước 1:** Chuẩn bị dữ liệu (dữ liệu đã được làm sạch, chuyển đổi, sẵn sàng đưa vào phân tích), chia tập dữ liệu ra làm 2: training dataset (để huấn luyện mô hình) và test dataset (để kiểm chứng mô hình).
- **Bước 2:** Chọn một số nguyên K bất kỳ, tức là số điểm dữ liệu đã được phân loại có khoảng cách gần nhất với điểm dữ liệu chưa phân loại.
- **Bước 3:** Tính toán khoảng cách giữa điểm dữ liệu chưa được phân loại với các điểm dữ liệu đã được phân loại.
- **Bước 4:** Với kết quả có được, sắp xếp theo thứ tự giá trị khoảng cách từ bé nhất đến lớn nhất.
- **Bước 5:** Chọn ra các điểm dữ liệu có giá trị khoảng cách bé nhất với điểm dữ liệu cần phân loại dựa trên K cho trước, ví dụ nếu K = 2 tức chọn ra 2 điểm dữ liệu gần nhất, K = 3 sẽ chọn ra 3 điểm dữ liệu gần nhất.
- **Bước 6:** Tiếp theo xem xét giá trị của biến mục tiêu (biến phân loại) của các điểm dữ liệu gần nhất, chọn ra giá trị xuất hiện nhiều nhất và gán cho điểm dữ liệu chưa phân loại, ví dụ K = 3 tức chọn 3 điểm dữ liệu gần nhất, trong đó có 2 điểm dữ liệu được phân loại là Đỏ, điểm còn lại được phân loại là Xanh thì điểm dữ liệu chưa phân loại lúc này sẽ được phân loại là Đỏ.




Hình 2.1: Phân loại nhãn cho biến mục tiêu. Nguồn: Internet.

Bước khó khăn nhất của thuật toán KNN, cần sự kinh nghiệm của nhà phân tích, đó chính là chọn K là bao nhiêu.




Hình 2.2: Chọn lựa tham số K trong KNN. Nguồn: Internet.

Trong trường hợp K là một số chẵn (2, 4, 6, …) và các điểm dữ liệu gần nhất có số lượng nhãn phân loại là như nhau ví dụ K = 2 thì có 1 điểm dữ liệu được phân loại là Đỏ và điểm còn lại là Xanh, thì khi đó điểm chưa phân loại có thể được phân loại là Đỏ hoặc Xanh đều được. Để tránh trường hợp này ta nên sử dụng K có giá trị là số nguyên dương lẻ.

### 2.1.3. Ứng dụng của thuật toán KNN

- Trong y tế: Xác định bệnh lý của người bệnh mới dựa vào dữ liệu lich sử của các bệnh nhân có cùng bệnh lý, có cùng đặc điểm đã được chữa khỏi trước đây, hay xác định loại thuốc phù hợp.
- Trong lĩnh vực ngân hàng: Xác định khả năng khách hàng chậm trả các khoản vay hay rủi ro tín dụng do nợ xấu dựa trên phân tính điểm tín dụng (credit score); xác định xem liệu giao dịch có hành vi phạm tội, lừa đảo không.
- Trong giáo dục: Phân loại học sinh theo hoàn cảnh, học lực để xem xét việc hỗ trợ gì cho những học sinh ví dụ như hoàn cảnh khó khan nhưng học lực lại tốt.
- Trong kinh tế nói chung: Giúp dự báo các sự kiện kinh tế trong tương lai, dự báo tình hình thời tiết trong nông nghiệp; xác định xu hướng thị trường chứng khoán để lên kế hoạch đầu tư thích hợp.

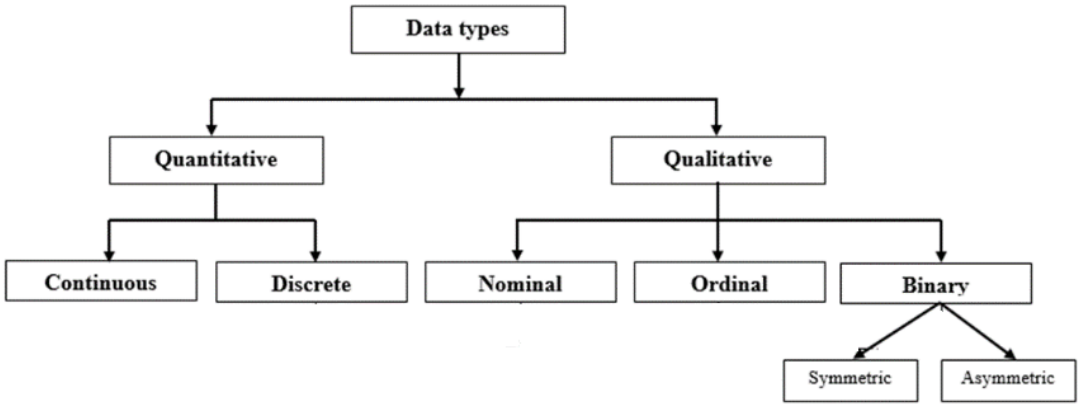
## 2.2. Thuật toán Logistic Regression[2]

### 2.2.1. Giới thiệu thuật toán

Dữ liệu (hay biến) thường có 2 loại chính là định tính (quanlitative/categorical variable), định lượng (quantitative/numberical variable), và nhị phân (binary variable).

- Biến định tính hay biến phân loại là biến phản ánh tính chất, hay loại hình, không có biểu hiện trực tiếp bằng con số ví dụ giới tính, nghề nghiệp, tình trạng hôn nhân. Có hai dạng Nominal (định danh) ví dụ nghề nghiệp, và Ordinal (thứ bậc) ví dụ thứ hạng (nhất, nhì, …).
- Biến định lượng là biến biểu hiện trực tiếp bằng con số ví dụ tuổi, chiều cao, trọng lượng. Biến định lượng được chia thành hai loại rời rạc (discrete) ví dụ số học sinh một lớp học và liên tục (continuous) ví dụ như nhiệt độ.

- Biến nhị phân là loại biến chỉ có 2 giá trị, 2 biểu hiện không trùng nhau của một đơn vị, nếu đơn vị không có giá trị này, thì phải chứa giá trị còn lại. Ví dụ có hoặc không, sống hoặc chết, 0 hay 1… . Biến nhị phân có 2 dạng: đối xứng (symmetric) và không đối xứng (asymmetric).




Hình 2.3: Hình mô tả các kiểu dữ liệu. Nguồn: Internet.

Với bộ dữ liệu Fruits-360 thì biến phân loại có thể coi là thuộc dạng biến nhị phân.

**Các loại biến hay dữ liệu của biến mục tiêu (biến phân loại) là cơ sở để lựa chọn phương pháp hồi quy tương ứng:**

- Với biến mục tiêu là biến định lượng liên tục thì phương pháp hồi quy có thể được sử dụng là Linear Regression (hồi quy tuyến tính) gồm có simple linear (tuyến tính đơn biến) và multiplt linear (tuyến tính đa biến). Ngoài ra còn có nhiều phương pháp phân tích hồi quy chuyên sâu khác.
- Với biến mục tiêu là biến định tính, hay biến nhị phân (hoặc biến rời rạc) thì phương pháp hồi quy chủ yếu và thường là duy nhất chính là Logistic Regression. Với biến định danh ta có phương pháp Nominal Logistic Regression, với biến thứ bậc ta có phương pháp Ordinal Logistic Regression, và Binary Logistic Regression cho biến nhị phân.

Như vậy Logistic Regression là phương phát hồi quy thông dụng nhất, áp dụng cho các biến mục tiêu không phải là biến định lượng liên tục. Biến nhị phân là dạng biến phổ biến nhất trong Logistic Regression, là dạng đầu tiên, và được giảng dạy phổ biến.

Logistic Regression chính là một thuật toán phân loại, đây là điểm khác biệt với Linear Regression. Hồi quy tuyến tính dự báo giá trị của biến mục tiêu dựa trên mối quan hệ giữa nó với các biến độc lập x. Còn hồi quy Logistic hướng đến việc dự báo xác suất, khả năng biến phụ thuộc (biến mục tiêu) y đạt một trong hai giá trị của nó dựa trên mối quan hệ giữa nó với các biến độc lập x.

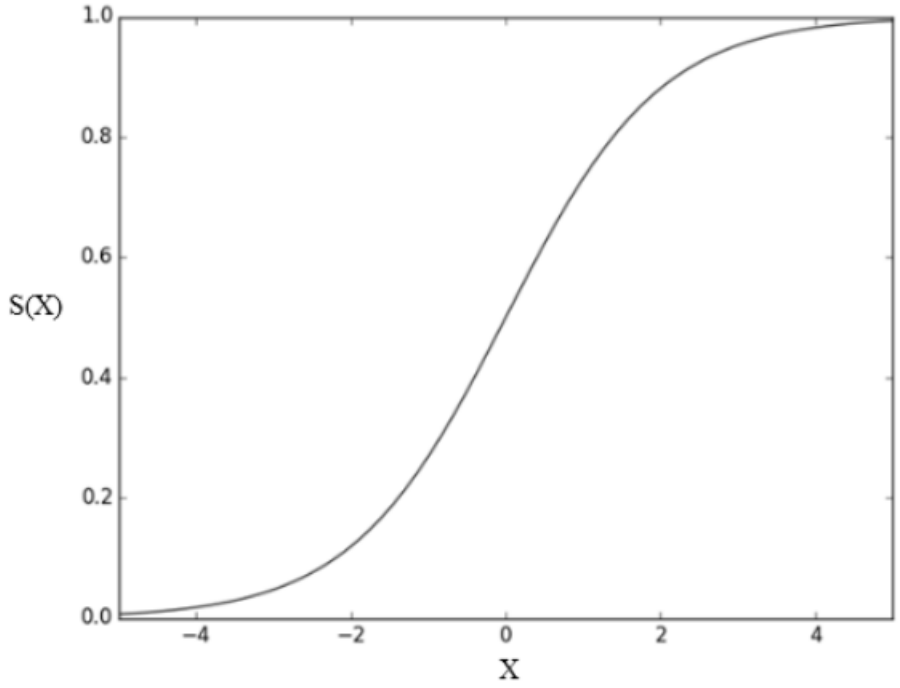
Vậy mô hình hồi quy logistic sẽ hướng đến tính toán, ước lượng xác suất để kết luận giá trị y chứ không phải được dùng để tìm ra giá trị sau cùng của y.

### 2.2.2. Mô tả cách thức vận hành thuật toán

Công thức tổng quát của hàm Sigmoid:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1} \quad \textbf{(1)}$$

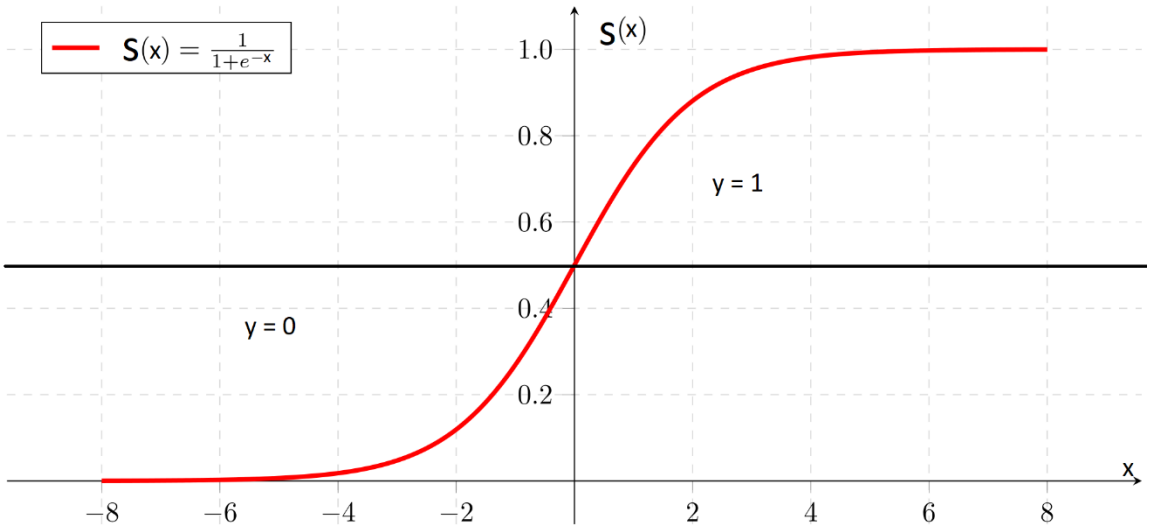
Đồ thị hàm Sigmoid:




Hình 2.4: Đồ thị hàm Sigmoid. Nguồn: Internet.

Hàm Sigmoid được nhận biết với đặc điểm nổi bật là đồ thị của nó có dạng hình cong như chữ S. Hàm Sigmoid thực chất bắt nguồn từ hàm Logistic. Vì thế mặc dù hàm Sigmoid được dùng trong mô hình hồi quy áp dụng cho biến mục tiêu là biến nhị phân nhưng không được gọi là Sigmoid Regression.

Trong hàm Sigmoid giá trị sau khi tính toán được chiếu thẳng lên điểm chính giữa của đường cong là mốc để phân loại, nghĩa là có thể lấy S(x) = 0.5 làm chuẩn.




Hình 2.5: Phân loại biến mục tiêu trong hàm Sigmoid. Nguồn: Internet.

Ta có thể dùng giá trị hàm Sigmoid **(1)** để phân loại như sau:

- $S(x) \geq 0.5$: biến mục tiêu y nhận giá trị phân loại là 1.
- $S(x) < 0.5$: biến mục tiêu y nhận giá trị phân loại là 0.

### 2.2.3. Ứng dụng của thuật toán Logistic Regression

- Dự báo hay phân loại email có phải spam hay không spam.
- Dự báo khả năng rời dịch vụ của khách hàng.
- Dự báo tình trạng khối u là lành tính hay ác tính trong y học.
- Dự báo khả năng khách hàng sẽ mua sản phẩm bất kỳ, hay đăng ký dich vụ.
- Dự báo khả năng trả nợ của khách.

## 2.3. Thuật toán Support Vector Machine (SVM)[3]

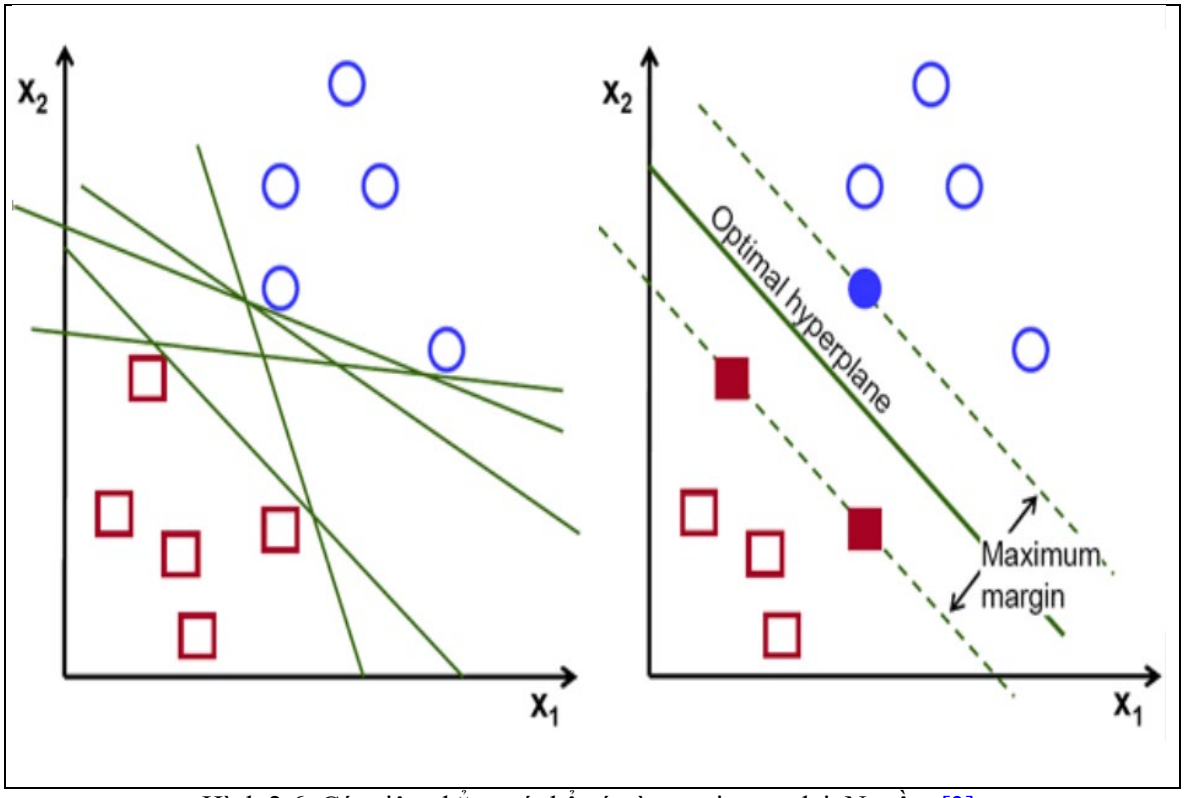
### 2.3.1. Giới thiệu thuật toán

SVM là một thuật toán cơ bản trong Machine learning. Được sử dụng rất nhiều bởi việc đưa ra độ chính xác cao mà không yêu cầu quá cao về việc tính toán.

SVM có thể được dùng cho bài toán hồi quy và cả bài toán phân loại. Trong đồ án này ta sẽ sử dụng SVM cho bài toán phân loại.

Mục tiêu của SVM là tìm ra một siêu phẳng (hyperplane) của không gian N-chiều (N là số lượng đặc trưng) điều này có tác dụng rất tốt trong việc phân loại các điểm dữ liệu.

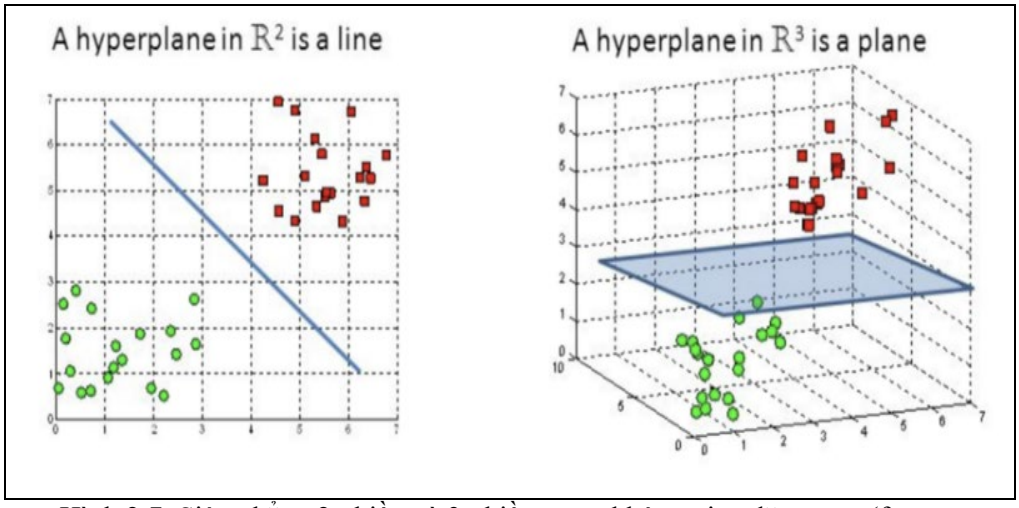
Để phân loại 2 nhãn của các điểm dữ liệu, có rất nhiều siêu phẳng có thể được tạo ra. Tuy nhiên mục tiêu của ta là tìm ra một siêu phẳng có lề (margin) cực đại. Việc cực đại biên độ giúp tạo ra một khoảng tin cậy vững chắc cho việc phân loại cái điểm dữ liệu mới trong tương lai.




Hình 2.6: Các siêu phẳng có thể có và margin cực đại. Nguồn: [3].

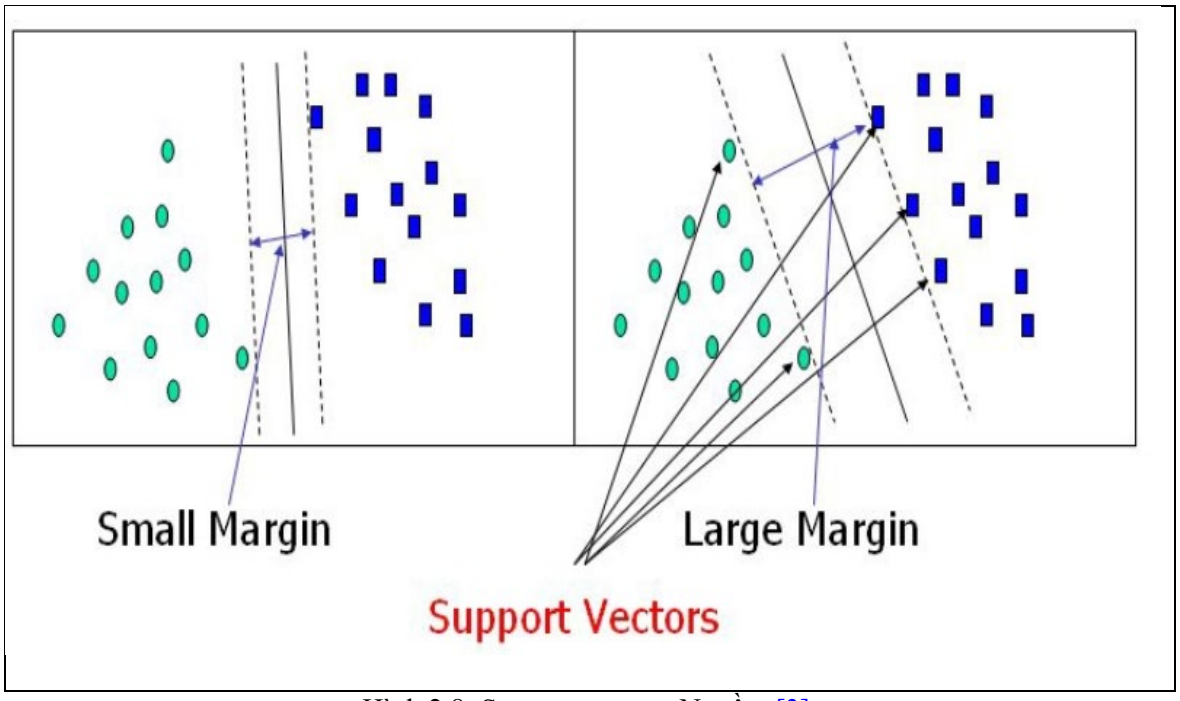
**Hyperplanes và support vectors:**

- **Hyperplanes** là các đường phân loại (decision boundaries) giúp ta phân loại các điểm dữ liệu. Các điểm dữ liệu nằm ở bên nào của siêu phẳng thì nhận giá trị phân loại tương ứng. Và chiều của siêu phẳng phụ thuộc vào số lượng đặc trưng. Nếu số lượng biến đặc trưng là 2 thì siêu phẳng là một đường thẳng. Nếu số lượng biến đăc trưng là 3 thì siêu phẳng sẽ là một mặt phẳng.




Hình 2.7: Siêu phẳng 2-chiều và 3-chiều trong không gian đặc trưng (feature space). Nguồn: [3].

- **Support vectors** là các điểm dữ liệu nằm gần siêu phẳng hơn so với các điểm dữ liệu khác và có ảnh hưởng tới vị trí và góc của siêu phẳng. Sử dụng các support vectors này, giúp ta tìm được margin cực đại cho việc phân loại. Loại bỏ hay di chuyển các support vectors sẽ làm thay đổi vị trí của siêu phẳng. **Đây là điểm quan trọng trong việc xây dựng SVM.**




Hình 2.8: Support vectors. Nguồn: [3].

### 2.3.2. Mô tả cách thức vận hành thuật toán[4]

Bài toán xây dựng một mô hình phân loại SVM là bài toán đi tìm một siêu phẳng sao cho margin là lớn nhất (hay đây là một bài toán tối ưu). Đây là lý do vì sao SVM còn được gọi là Maximum margin classifier.

**Khoảng cách từ một điểm tới một siêu phẳng:**

- Trong không gian 2 chiều, khoảng cách từ một điểm có tọa độ ($x_0$, $y_0$) tới đường thẳng có phương trình $w_1x + w_2y + b = 0$ được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{{w_1}^2 + {w_2}^2}} \quad \textbf{(2)}$$

- Trong không gian 3 chiều, khoảng cách từ một điểm có tọa độ ($x_0$, $y_0$, $z_0$) tới mặt phẳng có phương trình $w_1x + w_2y + w_3z + b = 0$ được xác định bởi:

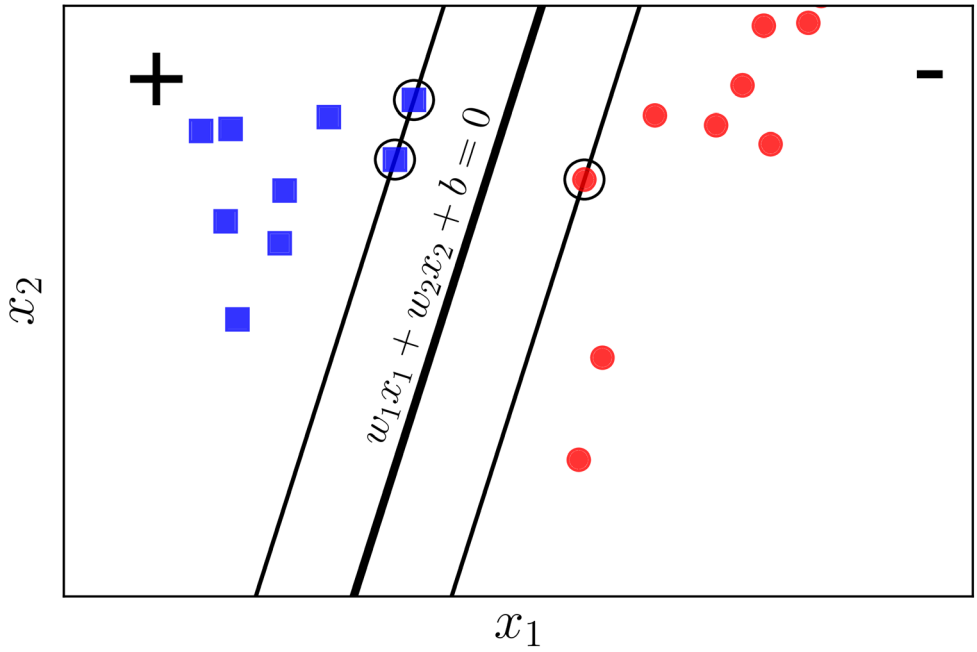
$$\frac{|w_1 x_0 + w_2 y_0 + w_3 z_0 + b|}{\sqrt{{w_1}^2 + {w_2}^2 + {w_3}^2}} \textbf{ (3)}$$

- Nếu bỏ dấu trị tuyệt đối ở tử số, ta có thể xác định được điểm đó nằm về phía nào của đường thẳng hay mặt phẳng đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu dương nằm về cùng 1 phía (tạm gọi đây là phía dương của đường thẳng), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (phía âm). Những điểm nằm trên đường thẳng/mặt phẳng sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.
- Việc này có thể được tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có toạ độ $x_0$ tới siêu phẳng có phương trình $w^T x_0 + b = 0$ được xác định bởi:

$$\frac{|w^T x_0 + b|}{||w||_2} \textbf{ với } ||w||_2 = \sqrt{\sum_{i=1}^{d} {w_i}^2} \textbf{ (4)}$$

*với d là số chiều của không gian*

Giả sử các cặp dữ liệu của tập huấn luyện là $(x_1, y_1)$, $(x_2, y_2)$, …, $(x_N, y_N)$ với vector $x_i$ thuộc $R^d$ thể hiện đầu vào của một điểm dữ liệu và $y_i$ là nhãn của điểm dữ liệu đó, với d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2).




Hình 2.9: Phân loại các nhãn trong SVM. Nguồn: [4].

Trong Hình 2.7 thấy rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộ class 2 và mặt $w^Tx + b = w_1x_1 + w_2x_2 + b = 0$ là siêu phẳng phân chia giữa 2 class. Và class 1 nằm về phía dương, class 2 nằm về phía âm của siêu phẳng. Nếu ngược lại, chỉ cần đổi dấu của w và b.

Có thể nhận thấy một điểm quan trọng: với cặp dữ liệu ($x_n$, $y_n$) bất kỳ, khoảng cách từ điểm đó tới siêu phẳng là:

$$\frac{y_i(w^T x_n + b)}{||w||_2} \quad \textbf{(5)}$$

Theo giả sử ở trên, $y_n$ luôn cùng dấu với phía của $x_n$. Từ đó suy ra $y_n$ cùng dấu với ($w^Tx_n + b$), và tử số luông là một số không âm.

Với siêu phẳng như Hình 2.7, margin được tính là khoảng cách gần nhất từ một điểm tới siêu phẳng đó (bất kể điểm nào trong hai class):

$$\textbf{margin} = \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{||\mathbf{w}||_2} \quad \textbf{(6)}$$

Bài toán tối ưu trong SVM chính là bài toán tìm w và b sao cho margin đạt giá trị lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\} \quad \textbf{(7)}$$

Khi tìm được cặp giá trị (w, b) phù hợp thì ta sẽ có được một mô hình phân loại SVM để phân loại các điểm dữ liệu.

Xác định class cho một điểm dữ liệu mới: sau khi tìm được siêu phẳng $w^Tx + b = 0$, nhãn của một điểm dữ liệu bất kỳ sẽ được xác định đơn giản:

$$\text{class}(x) = \text{sgn}(w^Tx + b) \quad \textbf{(8)}$$

Trong đó hàm sgn là hàm xác định dấu, nhận giá trị 1 nếu đối số là không âm và -1 nếu ngược lại.

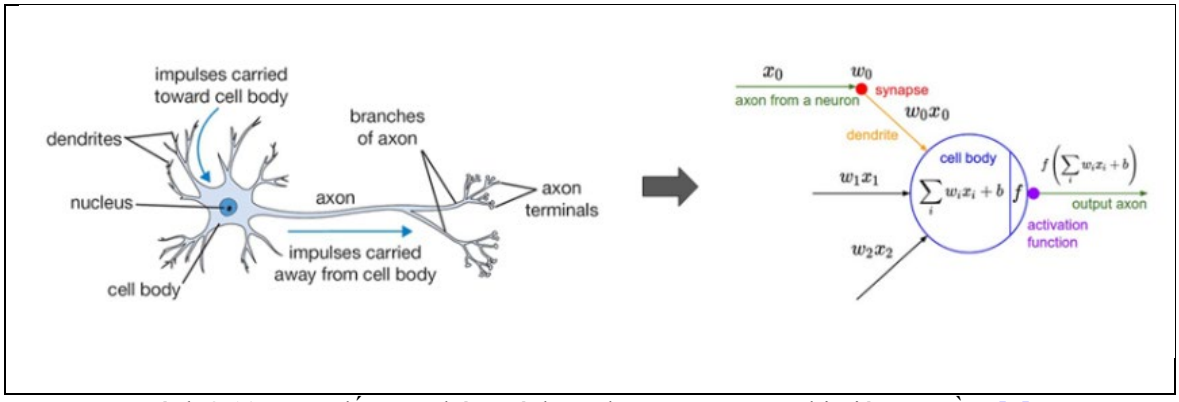
### 2.3.3. Ứng dụng của thuật toán SVM

- Nhận diện khuôn mặt (face detection): sử dụng SVM để phân loại cái thành phần trong ảnh là khuôn mặt người hay không phải khuôn mặt của người và tạo ra một khung bao quanh (square boundary) khuôn mặt.
- Phân loại văn bản: SVM cho phép phân loại văn bản (text) và siêu văn bản (hypertext) cho cả hai mô hình inductive và transductive. Sử dụng dữ liệu huấn luyện để phân loại cái tài liệu thành nhiều nhãn khác nhau. Nó phân loại trên cơ sở điểm số được tạo ra và sau đó so sánh với giá trị ngưỡng (threshold value).
- Phân loại ảnh: áp dụng SVM sẽ cho độ chính xác cao hơn trong việc phân loại ảnh. Nó cung cấp độ chính xác tốt hơn so với các kỹ thuật truy vấn truyền thống.
- Tin sinh học (bioinformatics): phân loại protein và phân loại ung thư.
- Nhận diện chữ số viết tay

## 2.4. Convolutional Neural Networks (CNNs)[5]

### 2.4.1. Giới thiệu thuật toán

Convolutional Neural Networks được lấy cảm hứng bơi quá trình phản ứng sinh học trong đó sự kết nối giữa các nơ-ron được truyền cảm hứng từ việc tổ chức của vỏ não thị giác (visual cortex) của động vật.




Hình 2.10: CNN lấy cảm hứng từ hoạt động của vỏ não thị giác. Nguồn: [5].

Trong tự nhiên, nơ-ron có một số lượng nhánh (inputs) lớn, một nhân tế bào (bộ xử lý) và một axon (output).

- Các nơ-ron là đơn vị cơ bản của một Neural Network.
- Chúng có thể liên kết với nhau, hay được dùng như những cổng kết nối giữa các nơ-ron khác.

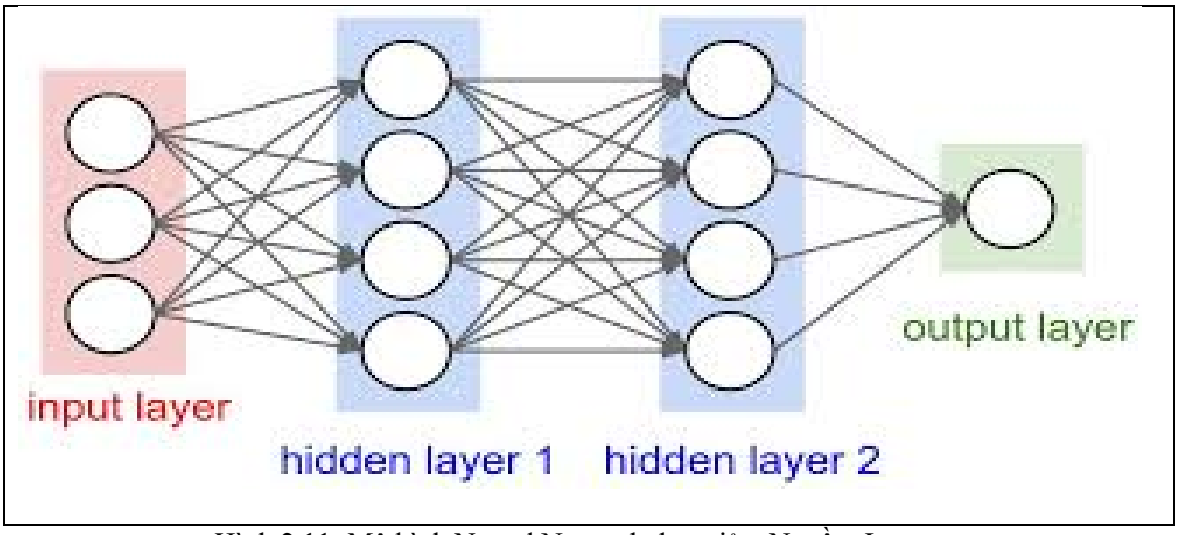
Một nơ-ron giống như một hàm, nó nhận vào một lượng inputs và trả về một output.

Khi một nơ-ron hoạt động, nó tích lũy tất cả các inputs truyền vào nó, và nếu tới một ngưỡng xác định thì nó sẽ phát ra một tín hiệu thông qua axon.

Điều quan trọng về nơ-ron là nó có thể học.

Một Neural Network được kết hợp với nhau bằng việc móc nối rất nhiều nơ-ron đơn với nhau, vì thế ouput của nơ-ron này có thể trở thành input của nơ-ron khác.

Một Neural Network gồm có một lớp input, vô số các lớp ẩn, và một lớp output. Mỗi node trong một lớp được kết nối với tất cả các node của lớp kế tiếp.

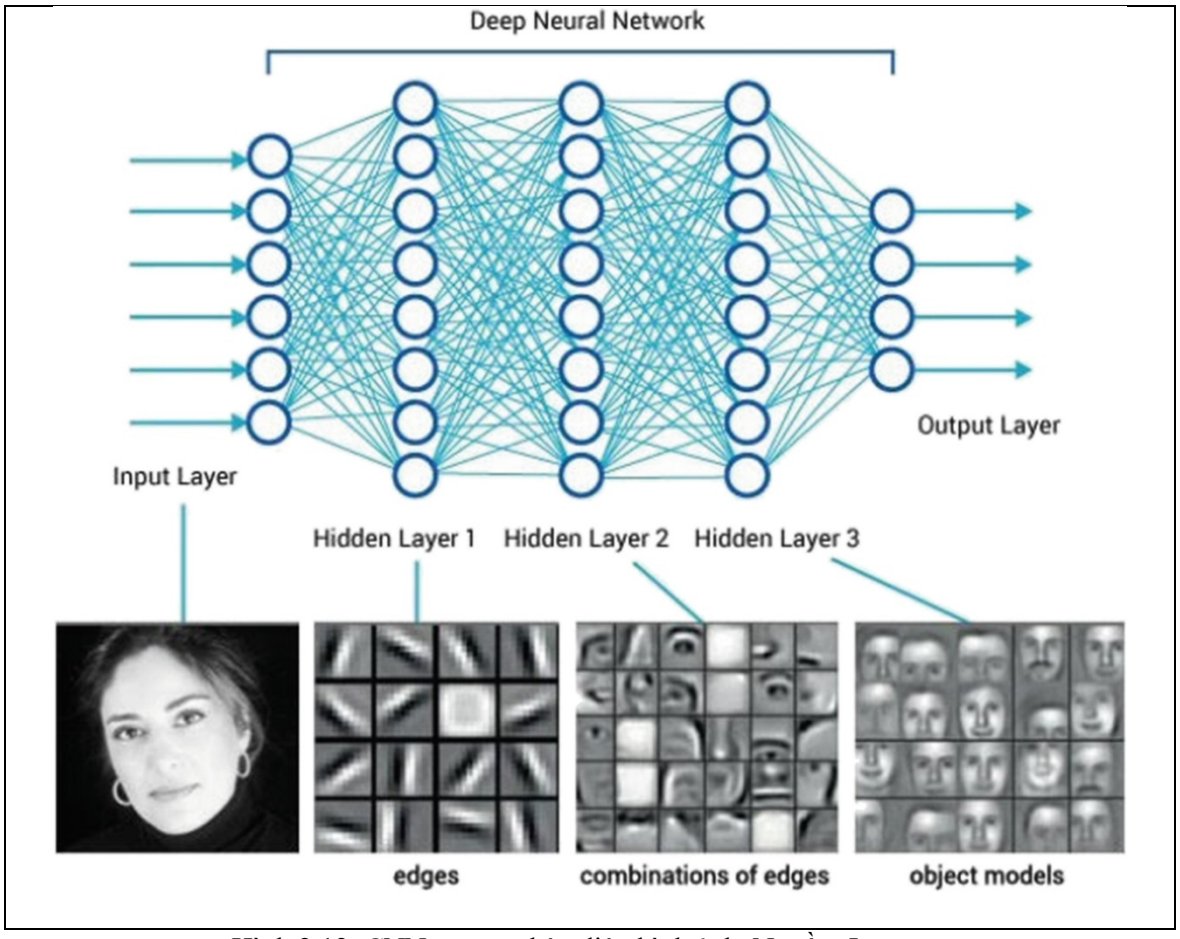



Hình 2.11: Mô hình Neural Network đơn giản. Nguồn: Internet.

Thành phần quan trọng nhất trong CNN là một lớp convolution. Nhiệm vụ của lớp này là xác định các đặc trưng quan trọng trong các pixel của ảnh. Những lớp gần với lớp input sẽ học để xác định các đặc trưng đơn giản như cạnh (edges) và độ dốc của màu (color gradients), ngược lại những lớp ở xa hơn sẽ tổng hợp các đặc trưng đơn giản này thành những đặc trưng phức tạp hơn. Cuối cùng những lớp dense ở phía cuối sẽ tổng hợp những đặc trưng ở mức độ level rất cao và đưa ra dự đoán phân loại (output).

### 2.4.2. Mô tả cách thức vận hành của thuật toán

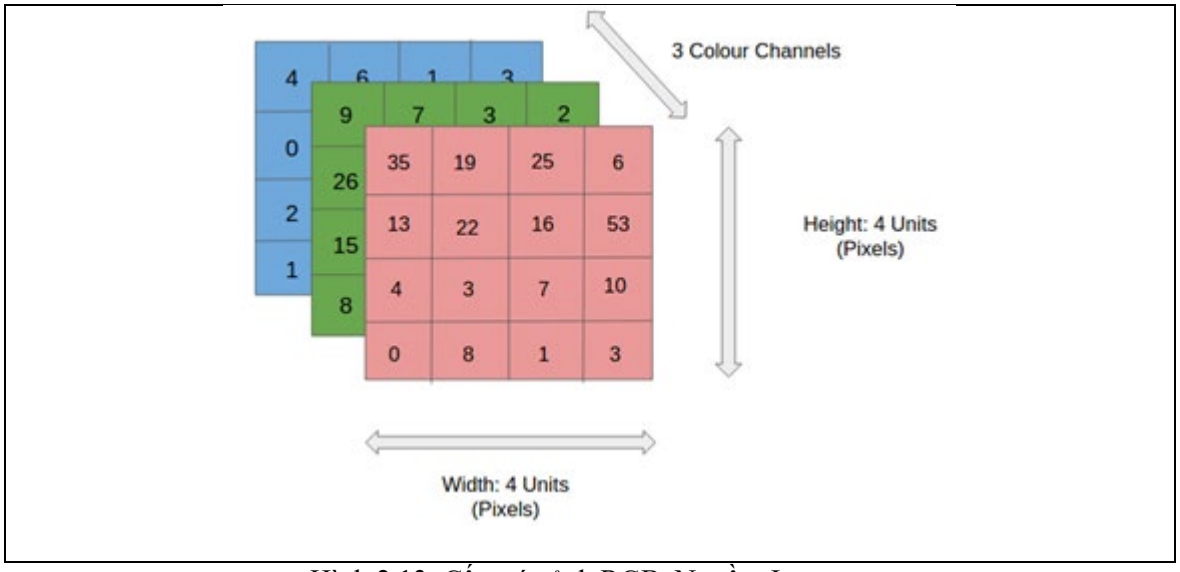
Convolutional Neural Networks được sử dụng để nhận diện các ảnh bằng cách chuyển đổi từ ảnh gốc, thông qua các lớp (layers), thành các chỉ số (score) của các lớp.




Hình 2.12: CNNs trong nhận diện hình ảnh. Nguồn: Internet.

#### 2.4.2.1. Input/Output của mô hình

CNNs thường được ứng dụng cho dữ liệu hình ảnh. Mỗi tấm ảnh là một ma trận pixel. Với những ảnh có màu, như ảnh RGB, có sự phân chia các kênh màu tạo ra thêm trường chiều sâu cho dữ liệu, tạo nên một input có 3 chiều. Do đó, với một ảnh RGB với kích thước 100x100 (chiều rộng x chiều cao) pixels, ta sẽ có 3 ma trận được tạo ra cho mỗi tấm ảnh, mỗi ma trận đại diện cho một kênh màu. Như vậy, một ảnh toàn diện được cấu thành bởi một cấu trúc 3 chiều gọi là Input volume (100x100x3).




Hình 2.13: Cấu trúc ảnh RGB. Nguồn: Internet.

#### 2.4.2.2. Đặc trưng (feature)

Một đặc trưng là một quan sát riêng biệt và hữu ích được lấy từ dữ liệu input, hỗ trợ rất nhiều trong việc phân tích hình ảnh. CNN học từ các đặc trưng của các ảnh đầu vào. Thông thường các đặc trưng xuất hiện lặp lại nhiều lần trong dữ liệu làm nổi bật lên một thông tin nào đó của ảnh.

#### 2.4.2.3. Bộ lọc (filter)

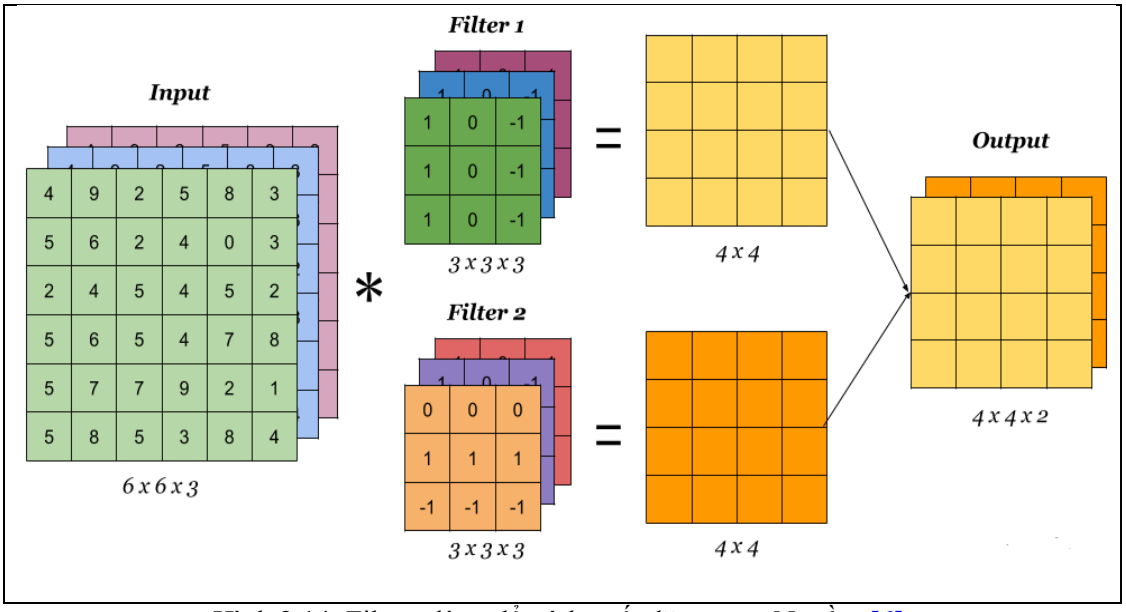
Filter (hay kernel) là thành phần không thể thiếu của kiến trúc lớp trong CNNs.

Bộ lọc giống như một dạng toán tử được áp dụng vào toàn bộ tấm ảnh để chuyển đổi thông tin được mã hóa trong các pixel.

Bộ lọc sau đó kết hợp với input volume để tạo ra một thành phần được gọi là activation maps (feature maps).

Activation maps đề cập đến các vùng được kích hoạt (activated), là các vùng mà các đặc trưng cụ thể được nhận diện bởi bộ lọc trong dữ liệu input.

Chiều của bộ lọc bằng với chiều của feature map.




Hình 2.14: Filters dùng để trích xuất đặc trưng. Nguồn: [6].

#### 2.4.2.4. Lớp kích hoạt (Activation layer)

Lớp kích hoạt là một hàm kích hoạt (activation function) dùng để đưa ra quyết định về giá trị cuối cùng của một nơ-ron. Giả sử như ta đặt ra một giá trị lý tưởng cho một cell (ô) nào đó có giá trị là 1, tuy nhiên nó lại có giá trị là 0.85, vì thế ta sẽ không thể đạt được giá trị xác suất là 1 trong CNN, do đó cần phải áp dụng hàm kích hoạt. Ví dụ như giá trị trong cell lớn hơn 0.7 thì sẽ chuyển thành 1 mà ngược lại ta sẽ chuyển thành 0.

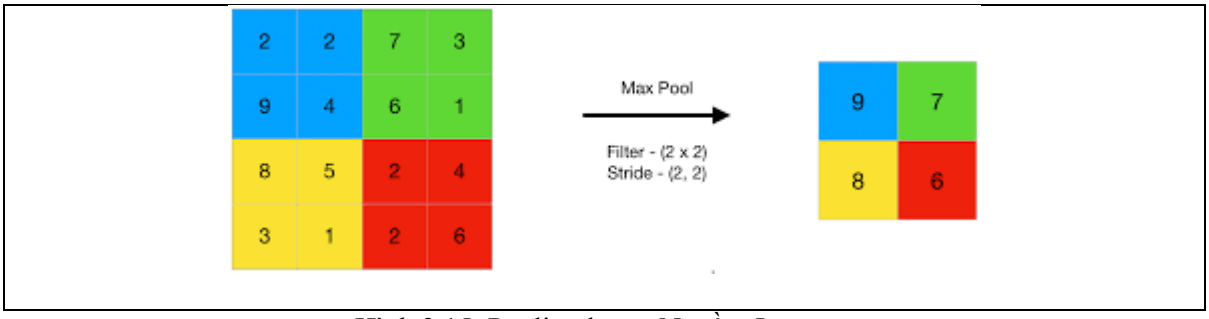
#### 2.4.2.5. Lớp Pooling (Pooling layer)

Lớp Pooling thường được đặt ngay sau lớp Convolution (Convolutional layer). Công dụng chính của lớp Pooling là giảm chiều không gian của Input volume cho Convolutional layer tiếp theo. Lớp pooling không làm thay đổi chiều sâu của dữ liệu (depth dimension).

Công việc của lớp Pooling còn được gọi là down-sampling, làm giảm kích thước đồng nghĩa với việc mất mát thông tin. Tuy nhiên sự hao hụt thông tin này lại có lợi vì:

- Giảm kích thước sẽ làm giảm chi phí tính toán cho các lớp phía sau.

- Giảm khả năng bị quá khớp (overfitting).




Hình 2.15: Pooling layer. Nguồn: Internet.

#### 2.4.2.6. Lan truyền ngược (Backpropagation)

Lan truyền ngược là quá trình ta cố gắng làm giảm lỗi (error) xuống mức thấp nhất có thể (error gần 0 nhất có thể).

Lỗi là hiệu của nhãn thực tế y với nhãn dự đoán y'. Quá trình này giúp tìm ra bộ trọng số w phù hợp nhất với tập dữ liệu đầu vào.

Ta thực hiện quá trình lan truyền ngược bằng cách sử dụng quá trình Gradient descent.

#### 2.4.2.7. Lớp fully connected (Fully connected layer)

Sau khi kết thúc lớp convolution và pooling, tiếp theo thường sẽ là các lớp fully connected, tại các lớp này mỗi pixel được xem như một nơ-ron riêng biệt giống như trong một Neural Network thông thường.

Lớp fully connected cuối cùng sẽ có số nơ-ron bằng đúng với số lượng nhãn cần dự đoán. Ví dụ như trong đồ án này, cần dự đoán 10 nhãn tương ứng với 10 loại trái cây, thì lớp fully connected cuối cùng sẽ có 10 nơ-ron tương ứng.

#### 2.4.2.8. Overfitting và Dropout

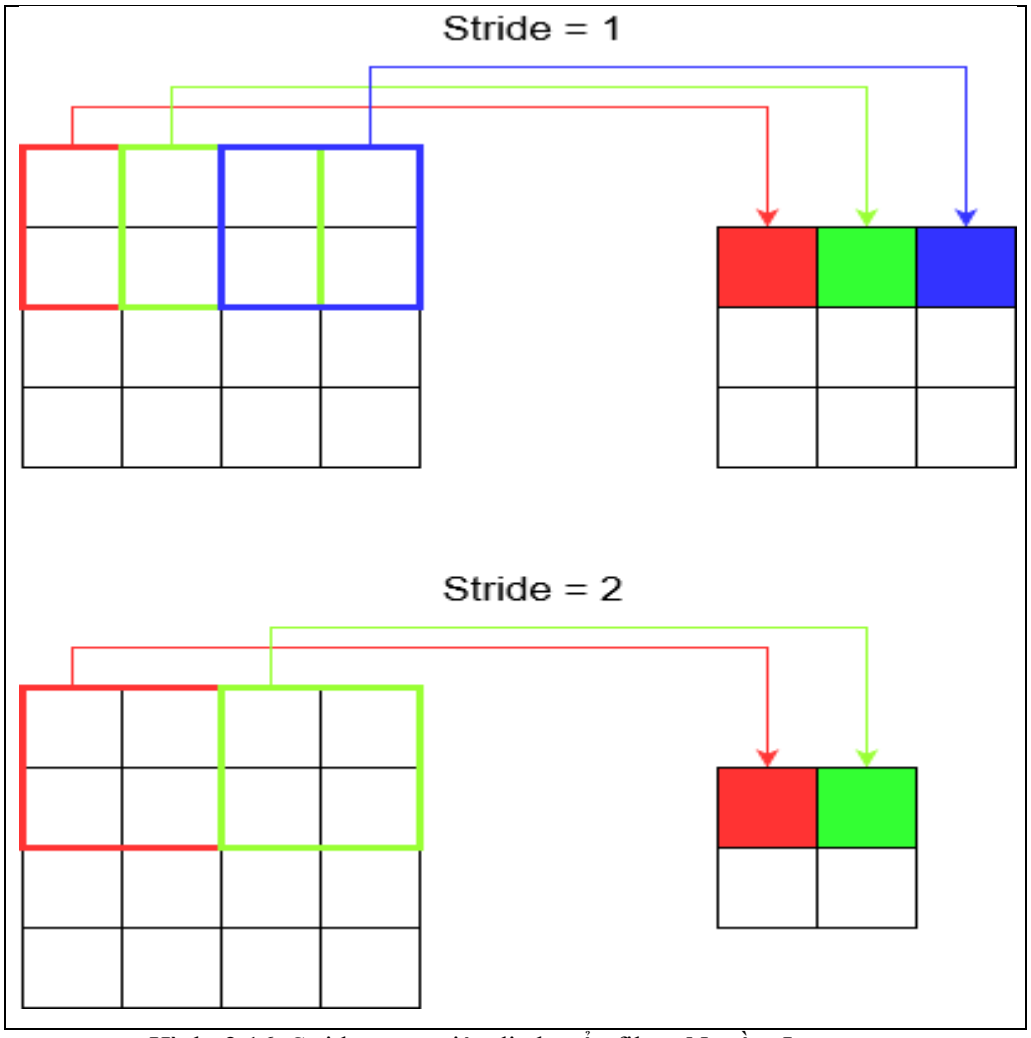
**Overfitting** là hiện tượng một mô hình có kết quả dự đoán rất tốt trên tập huấn luyện tuy nhiên lại không có kết quả dự đoán tốt trên tập kiểm thử.

**Dropout** là một kĩ thuật để giảm thiểu việc bị overfitting trong Neural Network.

#### 2.4.2.9. Depth, Stride và Zero-padding

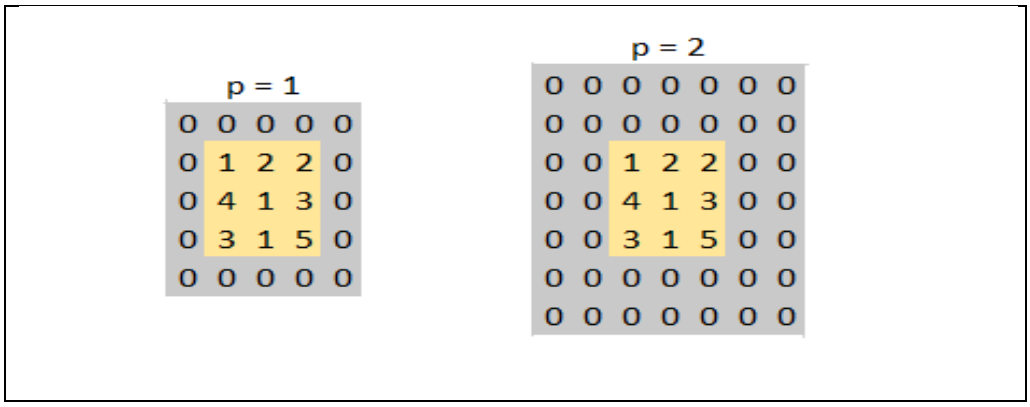
Có 3 tham số để điều khiển kích thước của output volume: **depth** (chiều sâu), **stride** (bước) và **zero-padding**.

- Depth (D) của output volume là một tham số, nó phụ thuộc vào số lượng filter mà ta sẽ dùng.
- Stride (S) với tham số này ta dùng để di chuyển filter trên ma trận của input volume. Với stride là 1 ta sẽ di chuyển filter sang một đoạn tương ứng 1 pixel. Khi stride là 2 ta sẽ di chuyển filter sang một đoạn tương ứng 2 pixel. Việc này sẽ tạo ra một output volume có kích thước không gian nhỏ hơn.

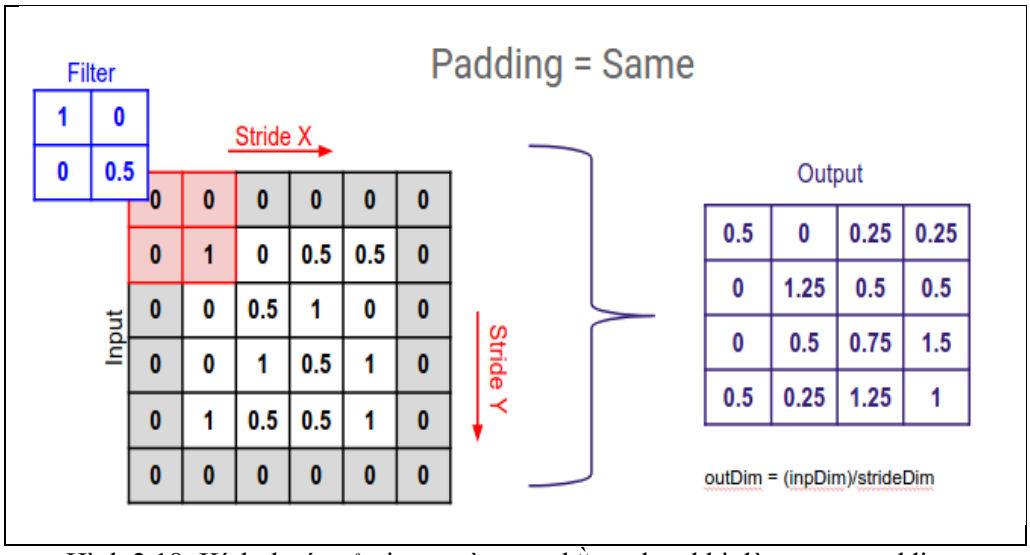



Hình: 2.16: Stride trong việc di chuyển filter. Nguồn: Internet.

- Zero-padding: là quá trình thêm số không một cách cân xứng vào ma trận đầu vào. Đây là một phép sửa đổi cho phép điều chỉnh kích thước của ma trận đầu vào phù hợp với yêu cần của ta. Trong một vài trường hợp việc này giúp đảm bảo được chiều của input volume sẽ bằng với chiều của output volume. Điều này cũng tránh việc filter bị trượt ra ngoài ma trận khi đang thực hiện lấy đặc trưng khi Stride lớn.

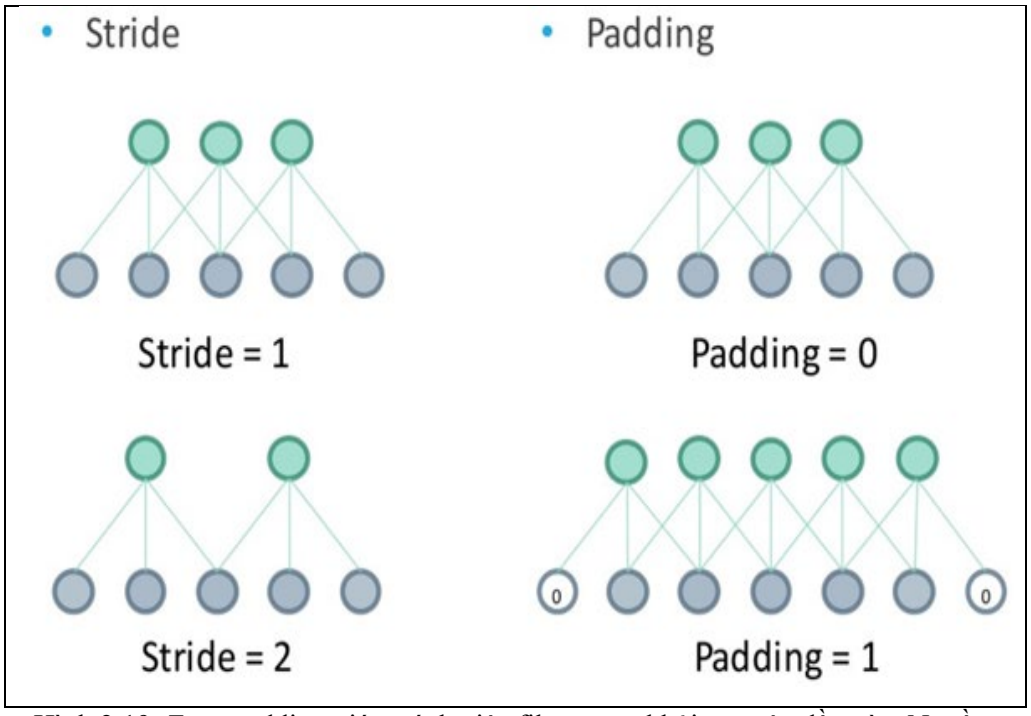



Hình 2.17: Zero-padding.




Hình 2.18: Kích thước của input và ouput bằng nhau khi dùng zero-padding.

Nguồn: Internet.




Hình 2.19: Zero-padding giúp tránh việc filter trượt khỏi ma trận đầu vào. Nguồn: [5].

#### 2.4.2.10. Công thức trong lớp Convolution

Nhận một input volume với kích thước **W1xH1xD1**.

Với 4 tham số bắt buộc:

- Số lượng filter: **K**
- Kích thước filter: **F**
- Độ dài Stride: **S**
- Số lượng Zero-padding: **P**

Tạo ra một output volume với kích thước **W2xH2xD2** với:

- **W2 = (W1 – F + 2*P)/S + 1**
- **H2 = (H1 – F + 2*P)/S + 1** **(9)**
- **D2 = K**

Với việc chia sẻ các tham số, dẫn đến sẽ có **F*F*D1** tham số cho mỗi filter, tổng số lượng tham số là **(F*F*D1)*K** và có **K** bias.

### 2.4.3. Ứng dụng của thuật toán CNN

CNN được ứng dụng trong rất nhiều lĩnh vực khác nhau bởi khả năng học rất tốt để có thể đưa ra những dự đoán có độ chính xác cao của nó:

- Nhận diện hình ảnh: khuôn mặt người, vật thể, động vật…
- Xây dựng các hệ thống khuyến nghị.
- Tạo ra chiến lược quảng cáo hiệu quả.

## 2.5. Phương pháp mô tả đặc trưng HOG[7]

### 2.5.1. Giới thiệu về HOG

HOG là một phương pháp mô tả đặc trưng (feature descriptor).

Mục đích của feature descriptor là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.

Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient (intensity gradient) hoặc của hướng biên (edge directions) để mô tả các đối tượng cục bộ trong ảnh. Các toán tử HOG được cài đặt bằng cách chia nhỏ một bức ảnh thành các vùng con, được gọi là "tế bào" (cells) và với mỗi cell, ta sẽ tính toán một histogram về các hướng của gradients cho các điểm nằm trong cell. Ghép các histogram lại với nhau ta sẽ có một biểu diễn cho bức ảnh ban đầu. Để tăng cường hiệu năng nhận dạng, các histogram cục bộ có thể được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một vùng lớn hơn cell, gọi là các khối (blocks) và sử dụng giá trị ngưỡng đó để chuẩn hóa tất cả các cell trong khối. Kết quả sau bước chuẩn hóa sẽ là một vector đặc trưng có tính bất biến cao hơn đối với các thay đổi về điều kiện ánh sáng.

### 2.5.2. Cách thức hoạt động[8]

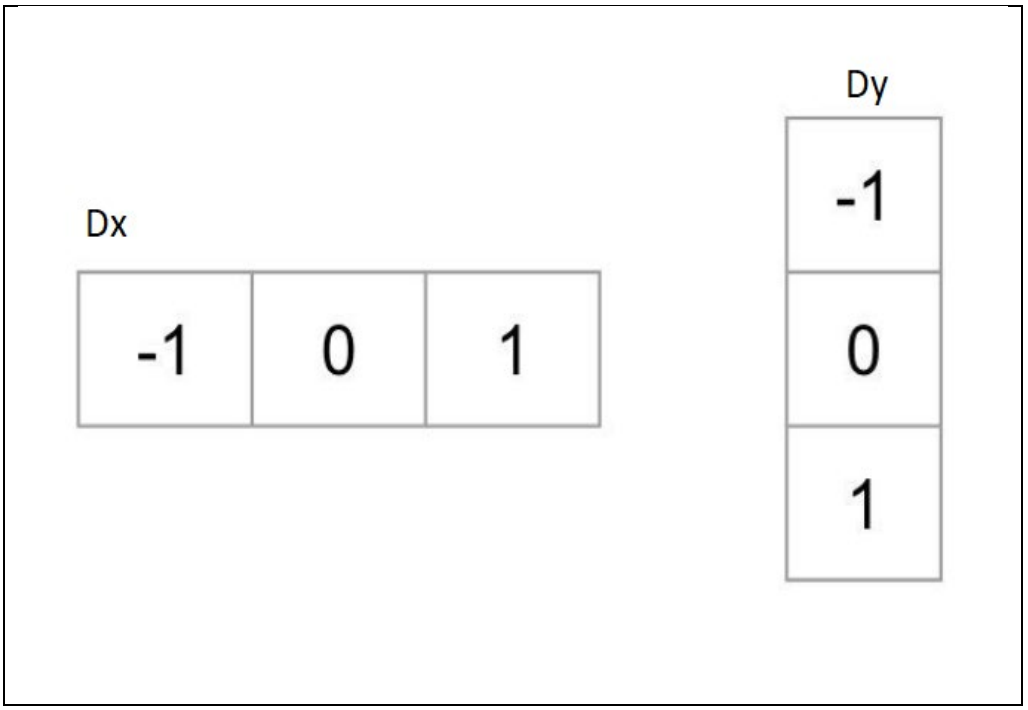
#### 2.5.2.1. Chuẩn hóa hình ảnh (global image normalisation)

Ở bước này, một bộ chuẩn hóa cân bằng ảnh được thiết kế để giảm thiểu ảnh hưởng của độ sáng. Trong thực tế, phương pháp gamma compression sẽ được sử dụng để tính căn bậc hai hay log của mỗi kênh màu. Sự liên kết trong tấm ảnh thì thường tỷ lệ với độ chiếu sáng bề mặt cục bộ vì vậy khi sử dụng phương pháp này giúp giảm thiểu ảnh hưởng của sự thay đổi của chiếu sáng và đổ bóng cục bộ trong hình ảnh.

#### 2.5.2.2. Tính toán các gradient của hình ảnh

Được thực hiện bằng hai phép nhân chập ảnh gốc với 2 chiều, tương ứng với các toán tử lấy đạo hàm theo hai hướng Ox và Oy. Giống như việc đạo hàm theo trục x và y là 2 tờ giấy hình chữ nhật A và B có cùng kích thước, khi đặt chồng lên nhau thì mỗi pixel trên tờ giấy A sẽ ứng với 1 pixel ở tọa độ tương ứng trên B.

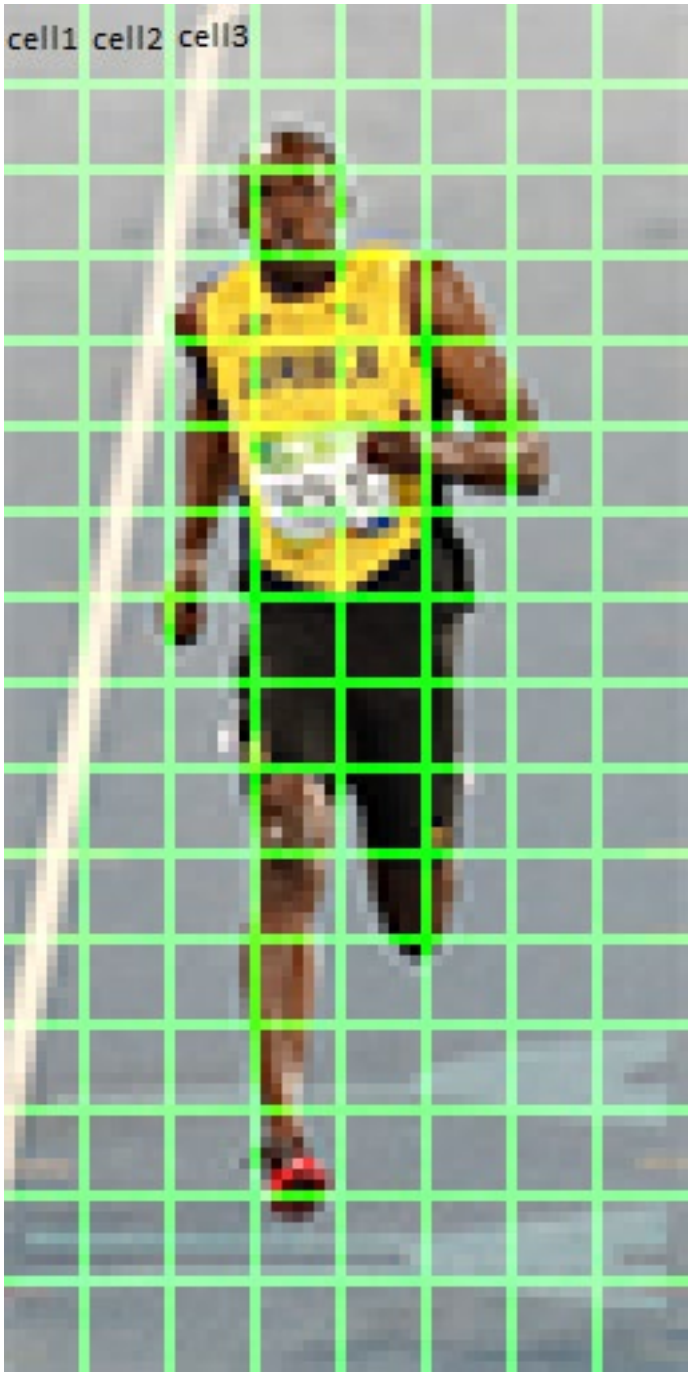
Có nhiều cách để tính gradient của ảnh (hay tính đạo hàm của ảnh), ta có thể dùng một filter 1-D đơn giản (-1, 0, 1) để convolution tính đạo hàm ảnh theo trục x, và chuyển vị của filter để tính đạo hàm ảnh theo trục y.



Hình 2.20: filter dùng tính đạo hàm ảnh.

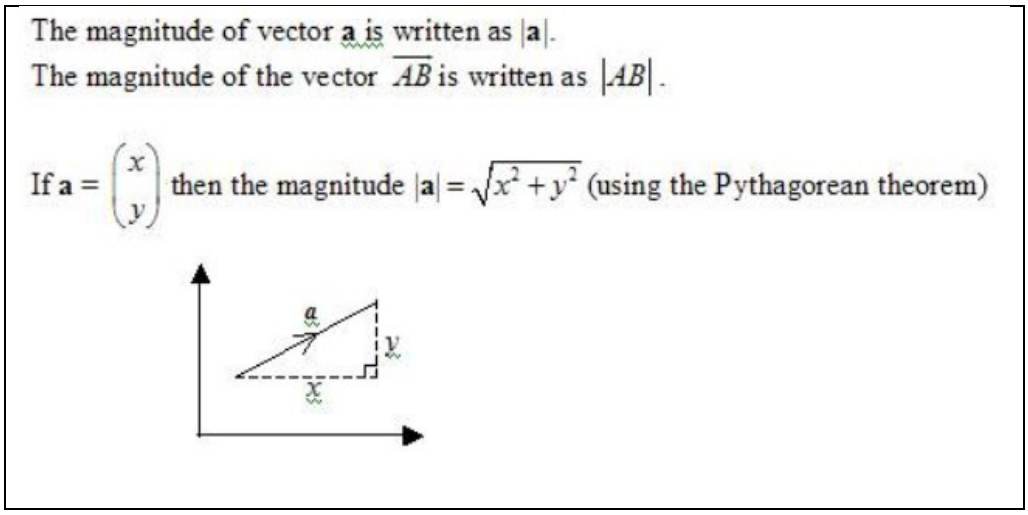
#### 2.5.2.3. Chọn hướng vào cells

cell được thiết kế có kích thước là 8x8 pixel (đây là siêu tham số). Do đó ảnh đầu có kích thước 64x128 sẽ có 8x16 = 128 cell (8 ô ngang và 16 ô dọc).




Hình 2.21: Chia các cell trong 1 tấm hình. Nguồn: [8].

Với cặp giá trị đạo hàm ảnh theo trục x và y thì ta sẽ tính được góc (hướng) và biên độ (độ lớn) tại pixel đang xét.

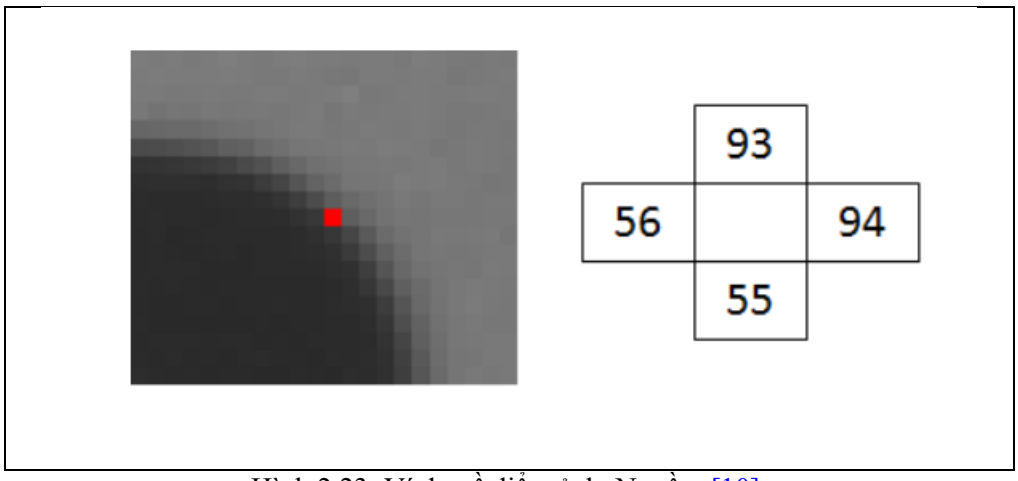



Hình 2.22: Công thức tính biên độ của ảnh. Nguồn: [9].

Dựa theo hình 2.22 ta có:

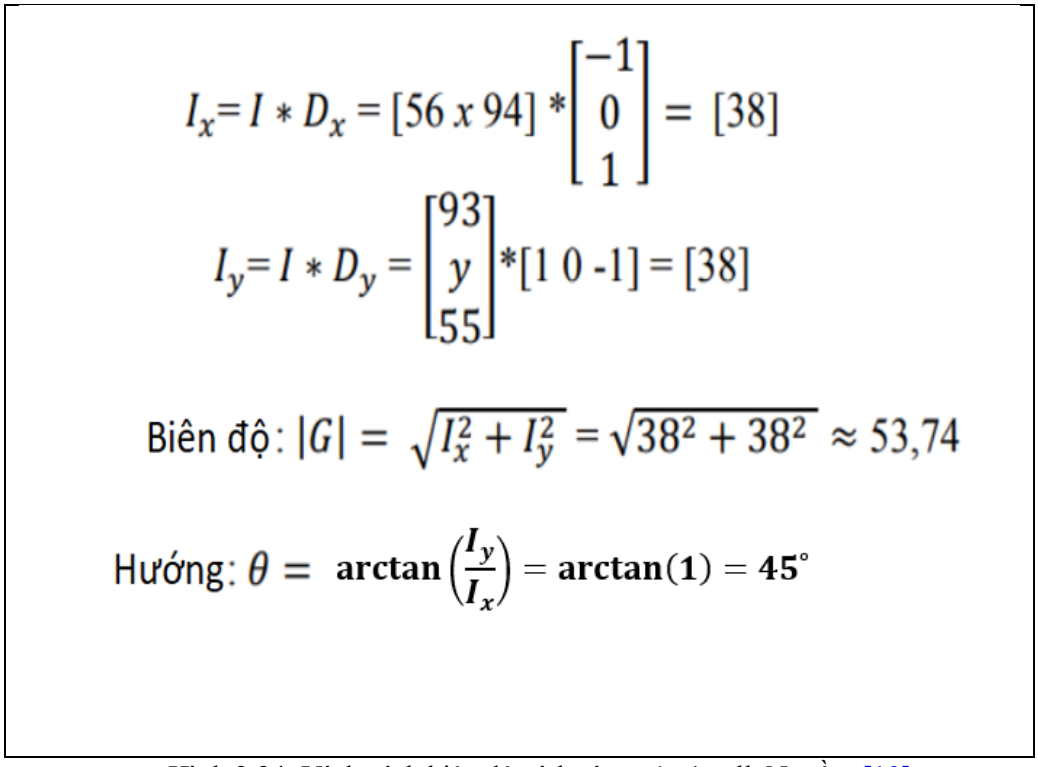
| | |
|---|---|
| Hướng = $\arctan(\frac{x}{y})$<br>Biên độ = $\sqrt{x^2 + y^2}$ | **(10)** |

Ví dụ: ta có một điểm ảnh như sau:




Hình 2.23: Ví dụ về điểm ảnh. Nguồn: [10].

Áp dụng công thức **(10)** ta tính được hướng và biên độ của cell:



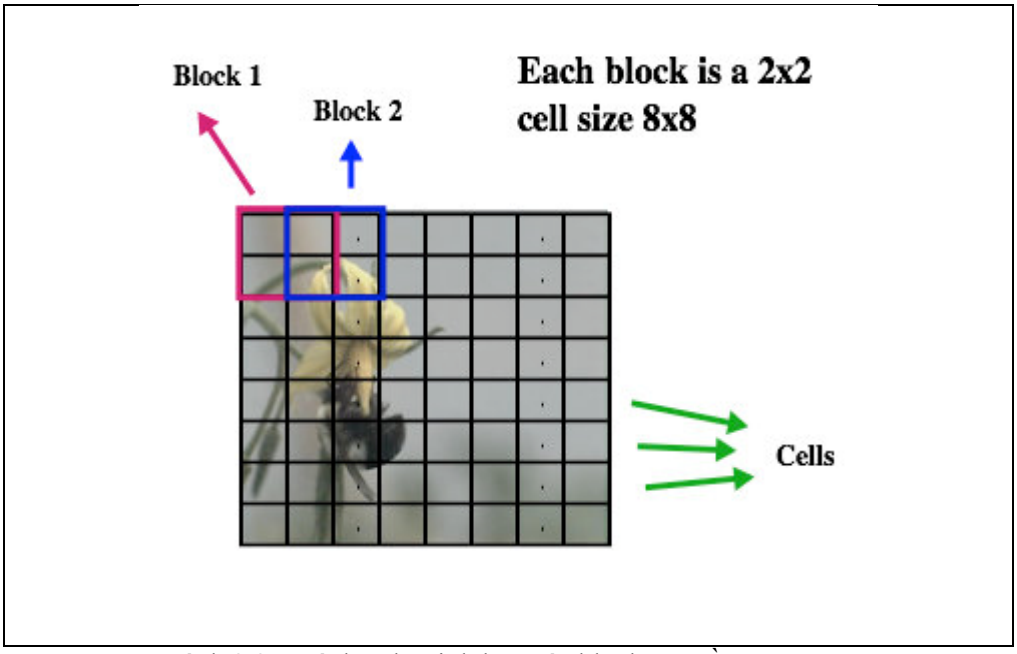

Hình 2.24: Ví dụ tính biên độ và hướng của 1 cell. Nguồn: [10].

Tiếp đến, ta xét lần lượt mỗi cell. Một cell kích thước 8x8 sẽ có 64 giá trị hướng và 64 giá trị biên độ trong cell đó. Ta sẽ tiến hành chọn hướng vào cell với các lựa chọn góc nằm từ 0-180 độ (các góc giá trị âm sẽ được lấy trị tuyệt đối quy về 0-180). Trong cung tròn 0-180 độ này, ta chia chúng thành các 9 đoạn rời rạc (9 bin). Nếu hướng thuộc khúc nào thì ta chọn vào bin đó. Quá trình này gọi là tính toán/ thống kê Histogram.

- Hướng 0-20 độ: bin 0
- Hướng 20-40 độ: bin 1
- Hướng 40-60 độ: bin 2
- Hướng 60-80 độ: bin 3
- Hướng 80-100 độ: bin 4
- Hướng 100-120 độ: bin 5
- Hướng 120-140 độ: bin 6
- Hướng 140-160 độ: bin 7
- Hướng 160-180 độ: bin 8

### 2.5.2.4. Chuẩn hóa các khối (blocks)

Một block gồm nhiều cell, block 2x2 nghĩa là ta có vùng diện tích của 4 cell liền kề –> block này sẽ phủ trên diện tích bằng 16x16 pixel. Trong quá trình chuẩn hóa, ta sẽ lần lượt chuẩn hóa block 2x2 đầu tiên, rồi dịch block đó sang 1 cell và cũng thực hiện chuẩn hóa cho block này. Như vậy, giữa block đầu tiên và block liền kề đã có sự chồng lấn cell lẫn nhau (2 cell).




Hình 2.25: Hình ảnh minh họa các block. Nguồn: Internet.

Để tiến hành chuẩn hóa ta lấy tất cả vector của 4 cell trong block đang xét nối lại với nhau thành vector v. Vector v có 9 x 4 = 36 (mỗi cell có 9 bin) phần tử. Sau đó ta chuẩn hóa (tính toán lại vector v) theo công thức bên bên dưới:

$$L2 - norm, v \rightarrow \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}} \quad (11)$$

Bản chất của chuẩn hóa L1-norm, L2-norm đó là:

- L1-norm: sau chuẩn hóa, tổng các giá trị của những phần tử trong vector bằng 1.
- L2-norm: sau chuẩn hóa, độ dài vector bằng 1.

#### 2.5.2.5. Tạo ra vector đặc trưng HOG

Block kích thước 2x2 chuẩn hóa trên các block overlap, như vậy tổng cộng ta sẽ có 7x15 block. Mỗi block mang trong mình 4 cell (2x2), mỗi cell có 9 bin. Từ đó, nếu ta phẳng hóa toàn bộ đặc trưng của tất cả các block có trên ảnh 64x128 ta sẽ được 1 vector đặc trưng có: 7 x 15 x 4 x 9 = 3780 phần tử.

### 2.5.3. Ứng dụng

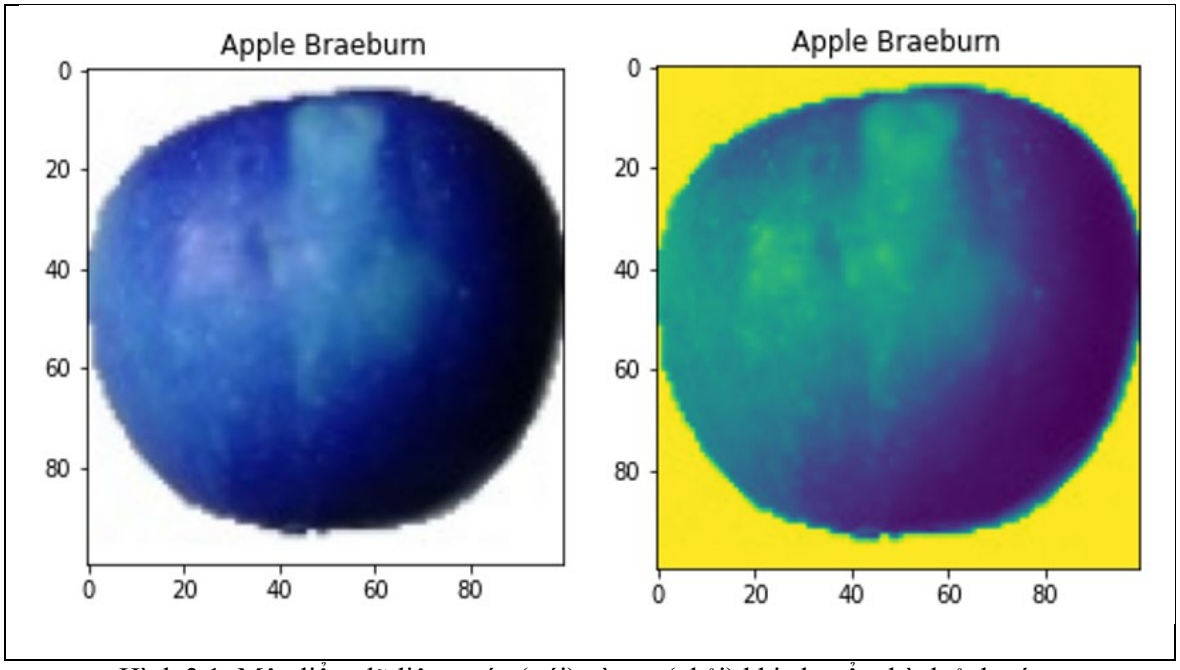
- HOG được ứng dụng nhiều trong bài toán nhận diện đối tượng trong ảnh.
- Giúp tối ưu cái thuật toán phân loại vì dùng đặc trưng HOG giúp loại bỏ các thông tin không quan trọng trong ảnh, chỉ còn lại những thông tin tối ưu nhất – đủ để phân biệt giữa 2 đối tượng khác nhau.

# Chương 3. QUÁ TRÌNH THỰC HIỆN

## 3.1. Tiền xử lý dữ liệu

### 3.1.1. Chuyển đổi dữ liệu thành ảnh xám

Sử dụng thư viện *CV2*[11] trong python để hỗ trợ chuyển ảnh gốc (RBG) sang ảnh xám




Hình 3.1: Một điểm dữ liệu trước (trái) và sau (phải) khi chuyển thành ảnh xám.

### 3.1.2. Chuẩn hóa dữ liệu

Thay đổi chiều của dữ liệu từ **WxHxD** thành **WxH**

- Chiều dữ liệu trước đổi chiều (ảnh xám): 100x100
- Chiều dữ liệu sau khi đổi chiều: 10000

Scale dữ liệu bằng cách sử dụng module *StandardScaler* được xây dựng trong thư viện *sklearn.preprcessing*.

- Dữ liệu sẽ được chuẩn hóa bằng cách loại bỏ trung bình (mean) và scale theo đơn vị phương sai (variance)
- Standard score của một mẫu (sample) x được tính bằng công thức:

$$Z = \frac{x-u}{s} \quad (12)$$

với u là trung bình của mẫu các dữ liệu huấn luyện (training samples) và s là độ lệc chuẩn của training samples.

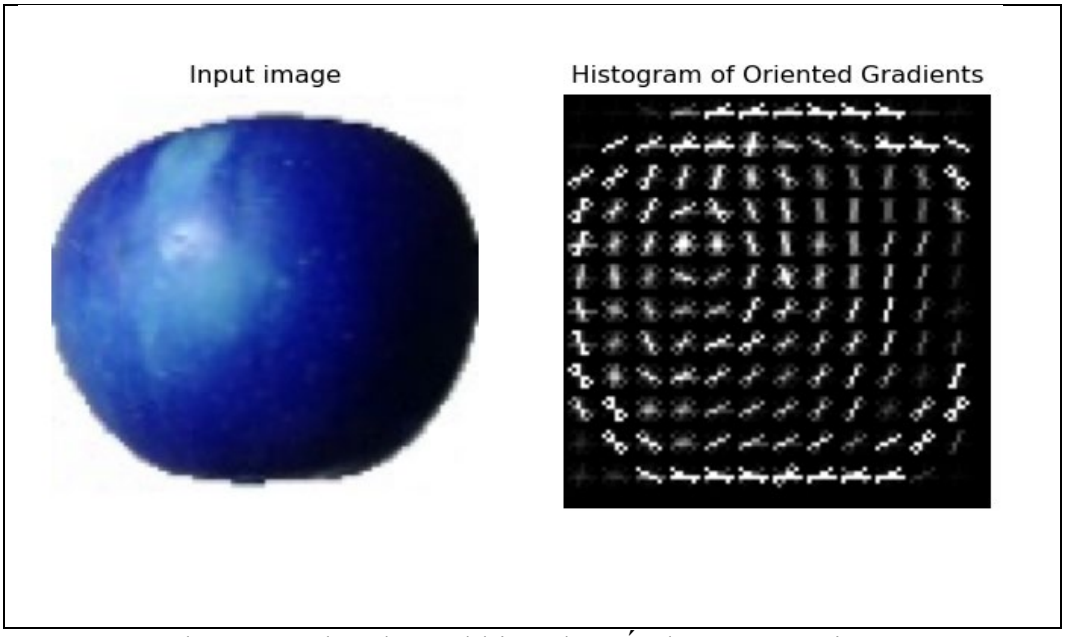
- Trung tâm hóa và scaling được thực hiện một cách độc lập trên mỗi đặc trưng bằng cách tính các thống kê liên quan trên các mẫu của tập huấn luyện. Giá trị trung bình và độ lệch chuẩn sau đó được lưu lại để dùng tiếp cho dữ liệu tiếp theo bằng cách sử dụng phương thức *transform.*

### 3.1.3. Trích xuất đặc trưng ảnh bằng HOG

Sử dụng *hog* đã được định nghĩ sẵn trong thư viện *skimage.feature*[12] để tiến hành chuẩn hóa hình ảnh theo phương pháp mô tả đặc trưng HOG.

Các thông số cài đặt trong *hog* như sau:

- Số lượng bin (orientations): 9
- Kích thức cell (pixels_per_cell): (8, 8)
- Số lượng cell trong mỗi block (cells_per_block): (2, 2)
- Phương pháp chuẩn hóa block (block_norm): L2




Hình 3.2: Hình ảnh sau khi trích xuất đặc trưng với HOG.

### 3.1.4. Đánh giá độ chính xác của mô hình

Sử dụng *confusion_matrix* và *accuracy_*score trong *sklearn.metrics*[13] để đánh giá độ chính xác của mô hình.

## 3.2. Cài đặt các mô hình

### 3.2.1. Mô hình KNN

Sử dụng *KNeighborsClassifier* được xây dựng trong *sklearn.neighbors*[14] để thực hiện cài đặt mô hình KNN.

Với mô hình KNN sẽ sử dụng 2 độ đo khoảng cách là *Minkowski* và *Euclidean* với số lượng neighbors là 3, 5, 7. Như vậy sẽ có 6 mô hình KNN.

Mục đích của việc sử dụng nhiều độ đo khoảng cách là để đánh giá mô hình tốt hơn, tìm ra được những thông số thích hợp của mô hình cho bộ dữ liệu.

### 3.2.2. Mô hình Logistic Regression

Sử dụng *LogisticRegression* được xây dựng trong *sklearn.linear-model*[15] để thực hiện cài đặt mô hình Logistic Regression.

Với các thông số thiết lập cho mô hình như sau:

- random_state: 0
- Số lần lặp tối đa để hội tụ (max_iter): 70000

### 3.2.3. Mô hình SVM

Sử dụng *SVC* được xây dựng trong *sklearn.svm*[16] để xây dựng mô hình SVM.

Với thông số cài đặt:

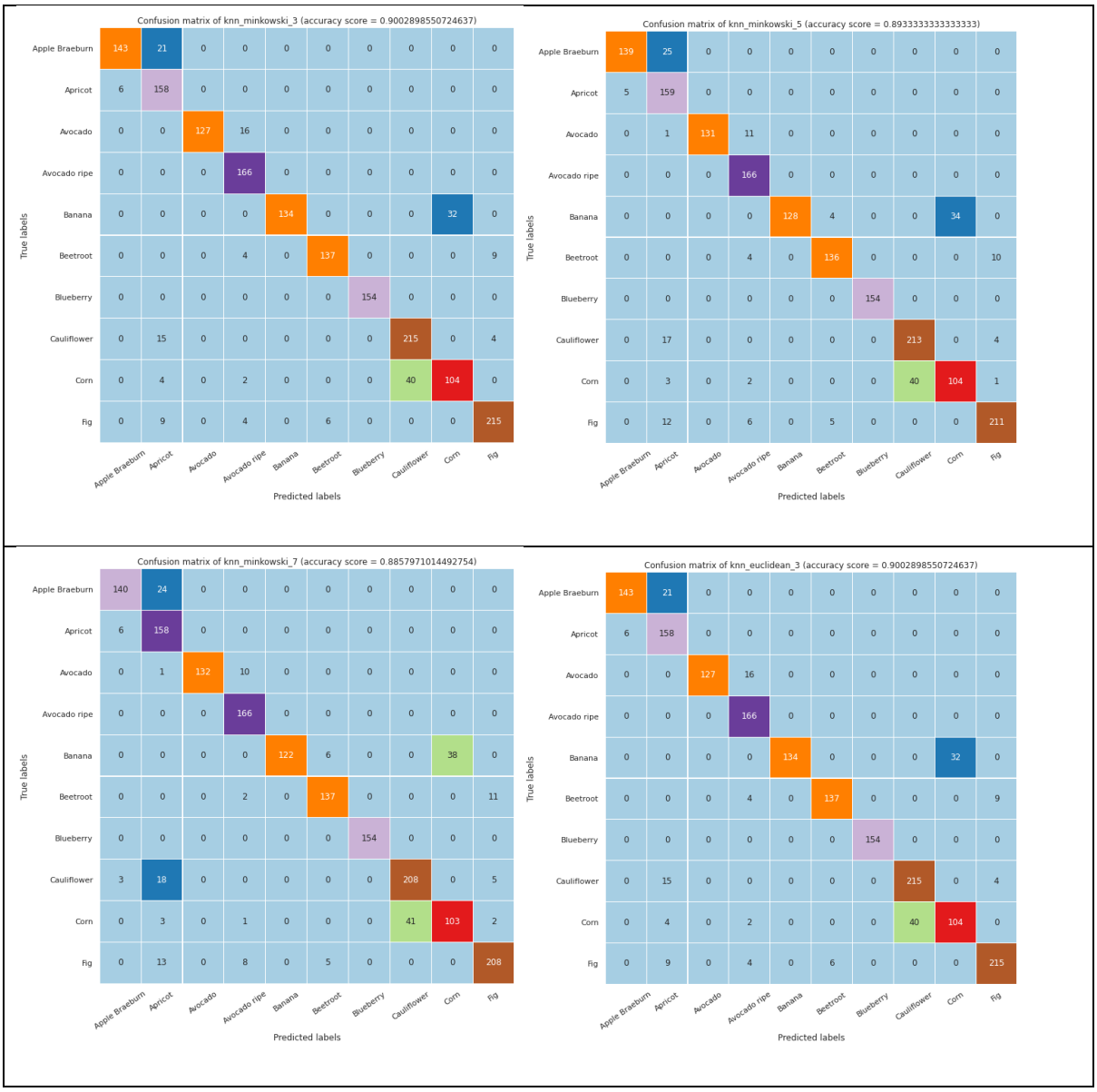
- Loại kernel: linear

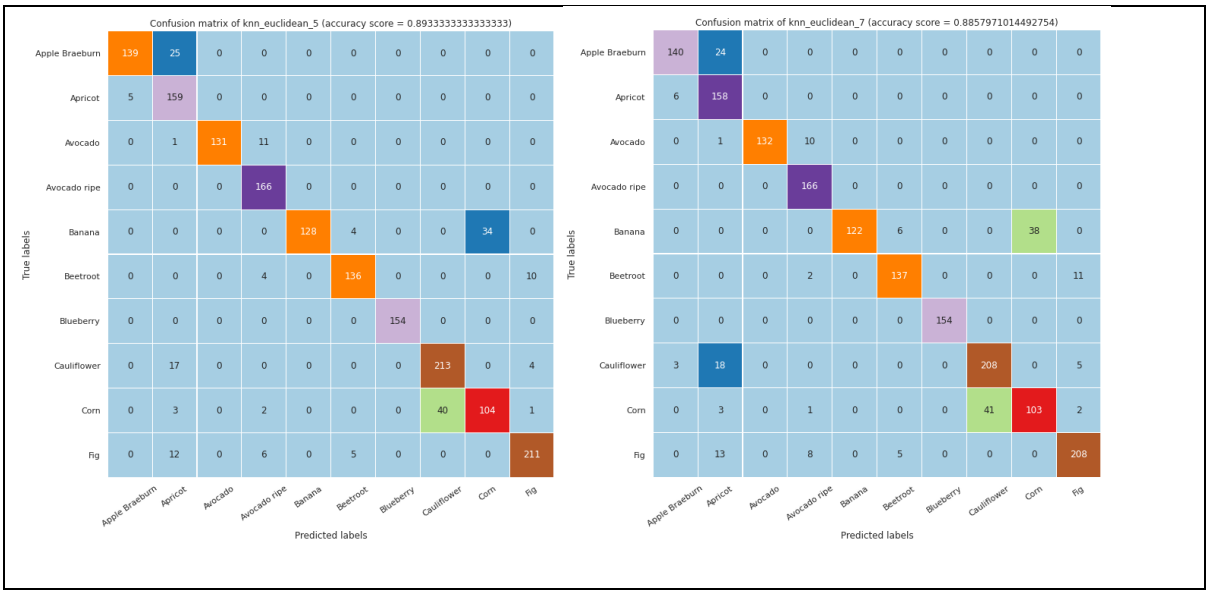
### 3.2.4. Mô hình CNN

Sử dụng *Model* được xây dựng trong *tensorflow.keras.models*[17] và *Input, Dense, Conv2D, MaxPooling2D, Flatten, Activation, Dropout, Lambda* trong *tensorflow.keras.layers*[18] để xây dựng mô hình CNN.

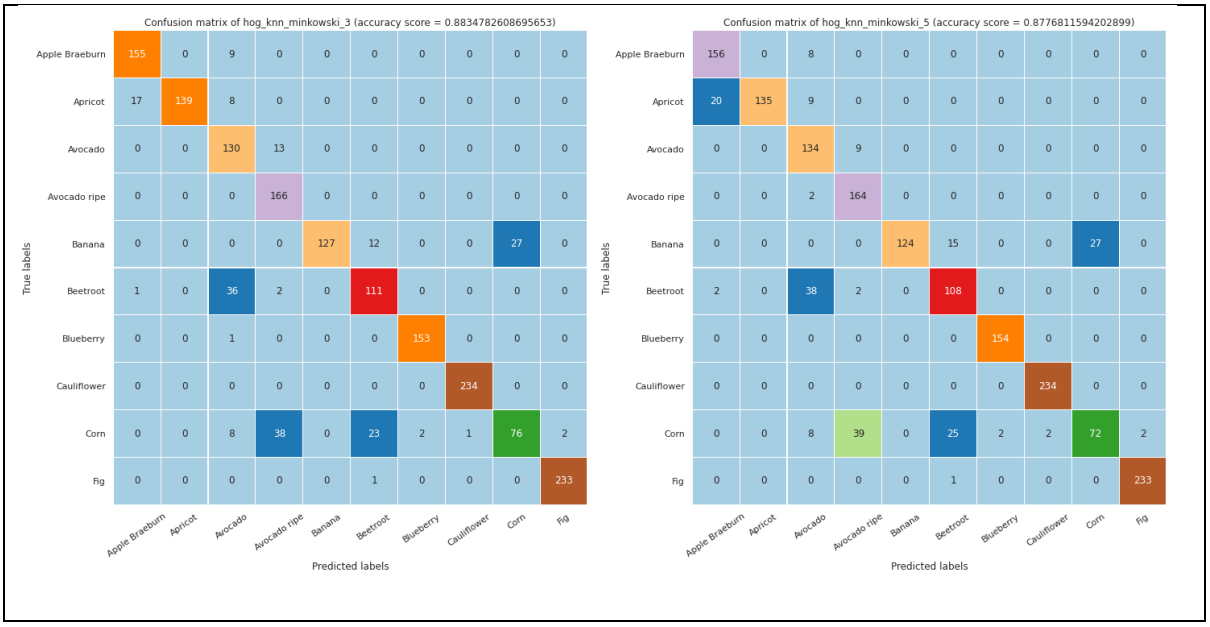
# Chương 4. **KẾT QUẢ**

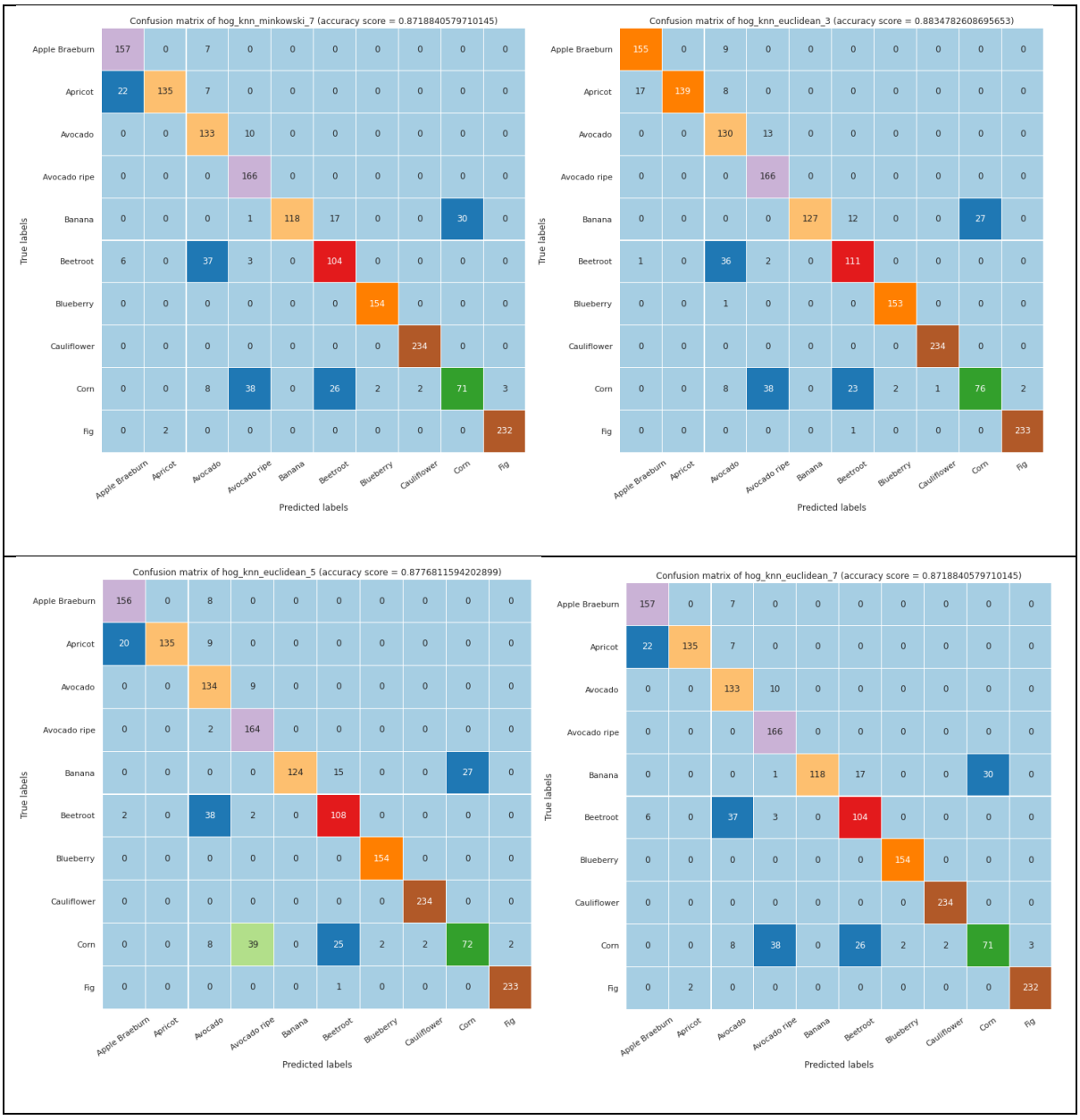
## 4.1. Kết quả dự đoán trên KNN





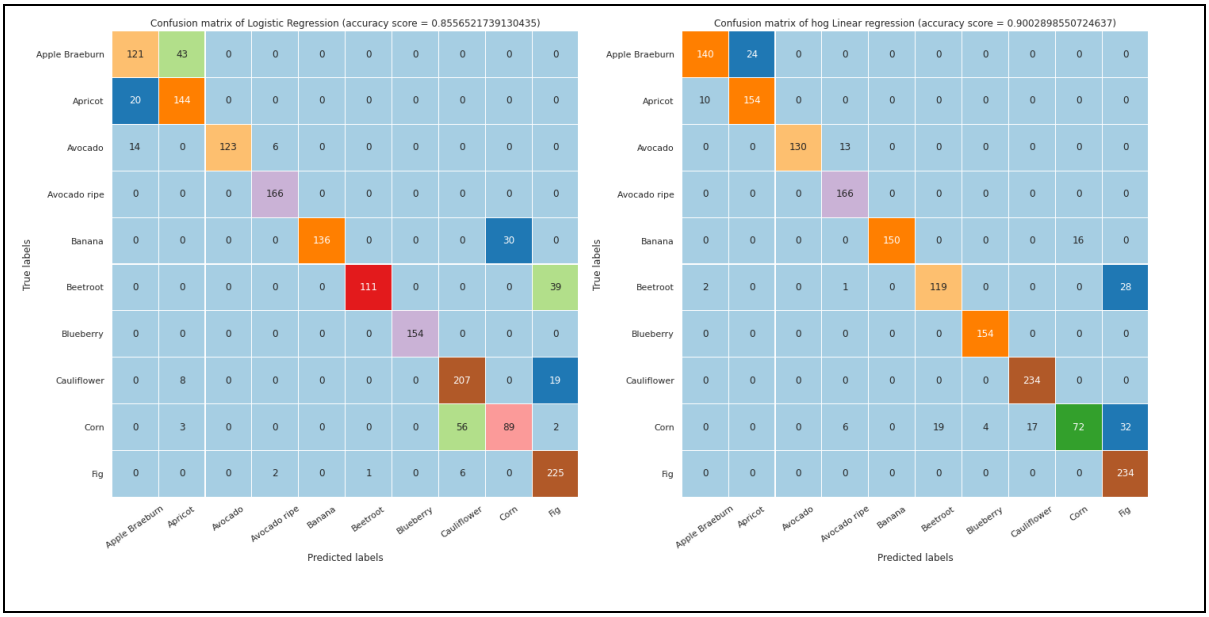

Hình 4.1: Các confusion matrix của các mô hình KNN.





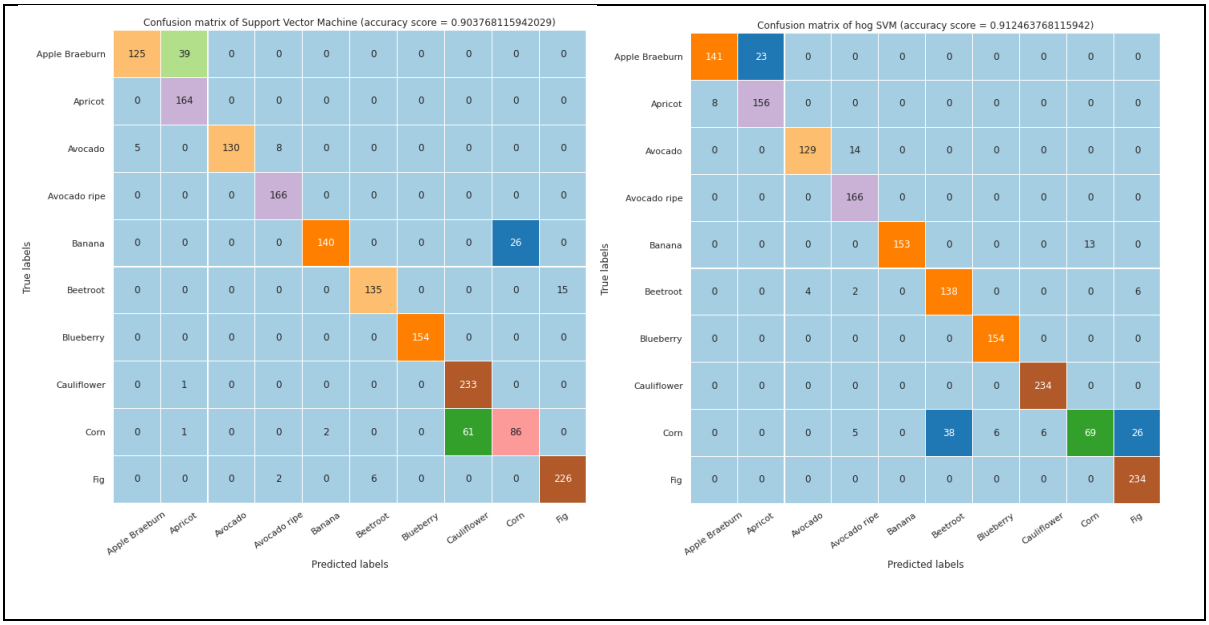

Hình 4.2: Các confusion matrix của các mô hình KNN có sử dụng HOG.

## 4.2. Kết quả dự đoán trên Logistic Regression



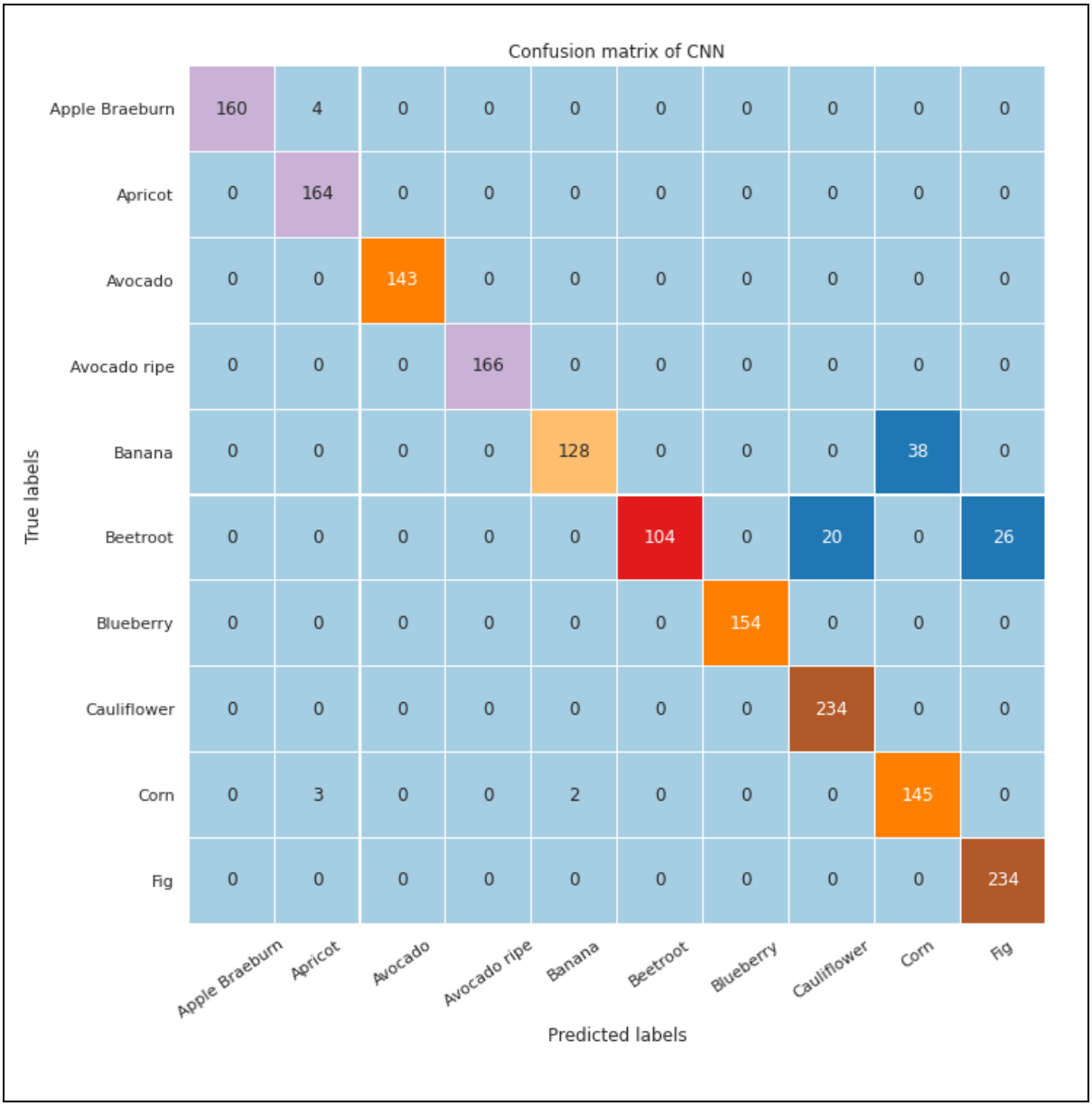

Hình 4.3: Các confusion matrix của mô hình Logistic Regression.

## 4.3. Kết quả dự đoán trên SVM




Hình 4.4: Các confusion matrix của mô hình SVM.

## 4.4. Kết quả dự đoán trên CNN




Hình 4.5: Confusion matrix của mô hình CNN.

## 4.5. Độ dự đoán chính xác trên các mô hình

| Mô hình | Độ chính xác (%) |
|---|---|
| Linear Regression | 85.6 |
| SVM | 90.4 |
| KNN_minkowski_3 | 90 |
| KNN_minkowski_5 | 89 |
| KNN_minkowski_7 | 88.6 |
| KNN_euclidean_3 | 90 |
| KNN_ euclidean_5 | 89 |
| KNN_ euclidean _7 | 88.6 |
| CNN | 93 |

Bảng 4.1: Độ chính xác của các mô hình dự đoán

| Mô hình | Độ chính xác |
|---|---|
| Logistic Regression | 90 |
| SVM | 91 |
| KNN_minkowski_3 | 88 |
| KNN_minkowski_5 | 87.8 |
| KNN_minkowski_7 | 87 |
| KNN_euclidean_3 | 88 |
| KNN_ euclidean_5 | 87.8 |
| KNN_ euclidean _7 | 87 |

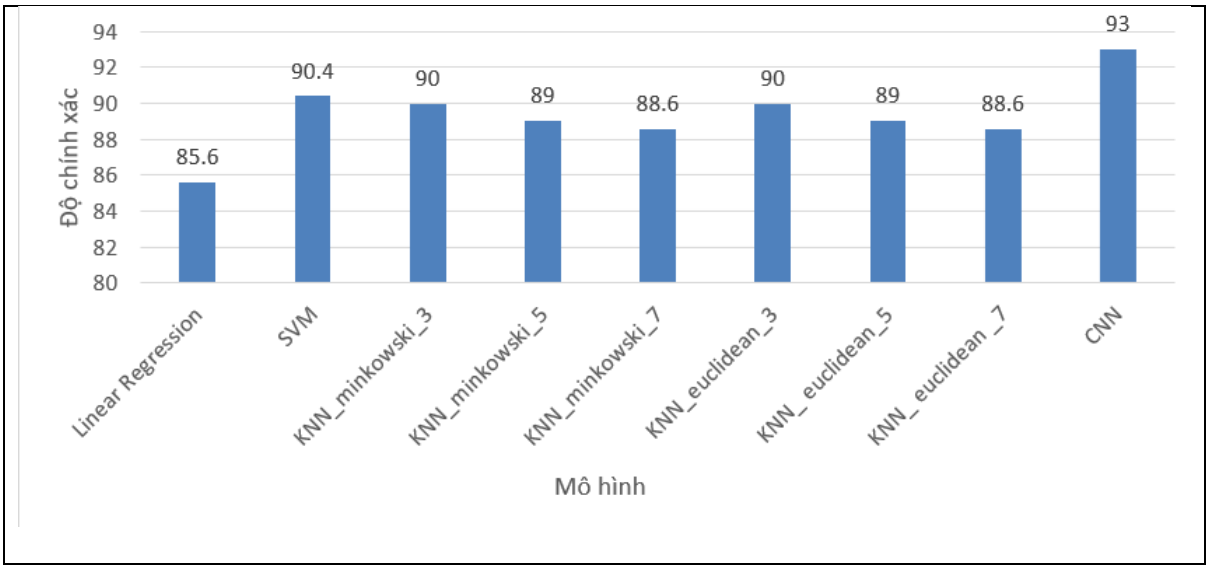
Bảng 4.2: Độ chính xác của các mô hình dự đoán có sử dụng HOG

# Chương 5. **KẾT LUẬN**

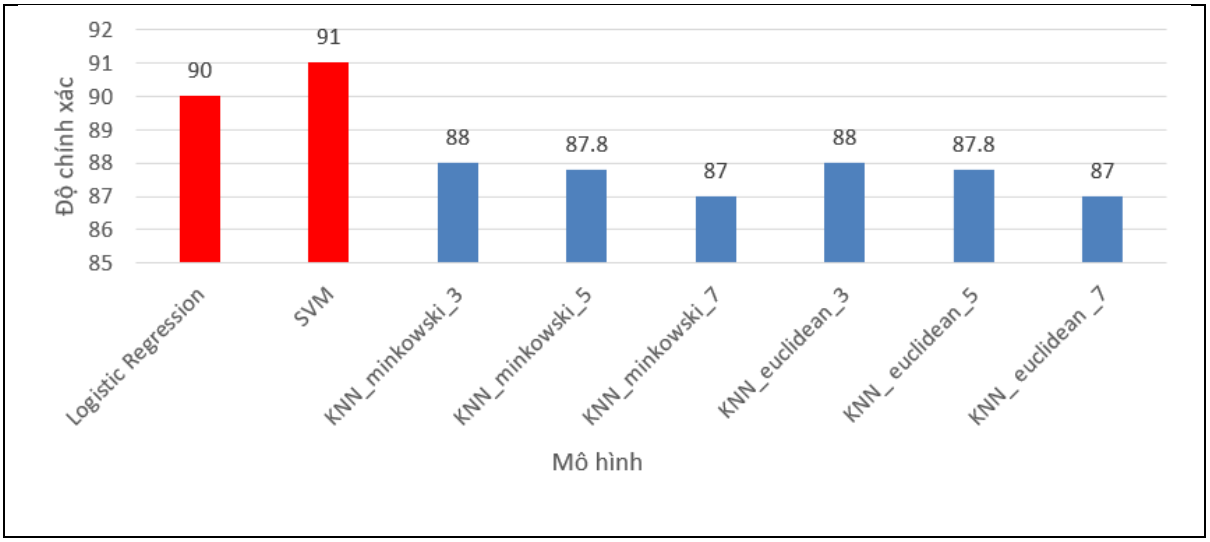
Việc sử dụng HOG giúp tăng độ chính xác của mô hình SVM và Linear Regression. Nhưng làm cho độ chính xác của mô hình KNN bị giảm.

Sử dụng 2 độ cho khoảng cách Minkowski và Euclidean cho kết quả dự đoán là như nhau với số lượng neighbors tương ứng.

Sử dụng mô hình CNN cho kết quả tốt nhất.




Hình 5.1: Độ chính xác của các mô hình




Hình 5.2: Độ chính xác của các mô hình có sử dụng HOG

## Phụ lục

| | |
|---|---|
| Source code | https://github.com/Cuan1407/Doanmonhoc_DS102.K21?fbclid=IwAR0QhcQJ2dPttYlHR6JWCLsFWsjQkmfOXjyXAe5VjBSfbdNwt83u1uPXUGI |
| Video | https://youtu.be/4VoR04ARn48 |

# TÀI LIỆU THAM KHẢO


[1] minhng, "HOG," [Online]. Available: https://minhng.info/tutorials/histograms-of-oriented-gradients.html. [Accessed 7 7 2020].

[2] big data uni, "Thuật toán KNN," big data uni, [Online]. Available: https://bigdatauni.com/vi/tin-tuc/thuat-toan-knn-va-vi-du-don-gian-trong-nganh-ngan-hang.html. [Accessed 10 7 2020].

[3] big data uni, "Logistic Regression," big data uni, [Online]. Available: https://bigdatauni.com/vi/tin-tuc/tong-quan-ve-logistic-regression-hoi-quy-logistic-phan-1.html. [Accessed 10 07 2020].

[4] Rohith Gandhi, "Support Vector Machine," [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed 8 07 2020].

[5] Vu Huu Tiep, "Support Vector Machine," [Online]. Available: https://machinelearningcoban.com/2017/04/09/smv/. [Accessed 8 07 2020].

[6] Raycad, "CNN," [Online]. Available: https://medium.com/@raycad.seedotech/convolutional-neural-network-cnn-8d1908c010ab. [Accessed 13 07 2020].

[7] ndoml.com, "CNN," [Online]. Available: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/. [Accessed 13 07 2020].

[8] sklearn, "HOG," [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html. [Accessed 7 7 2020].

[9] onlinemathlearning, "onlinemathlearning," [Online]. Available: https://www.onlinemathlearning.com/vector-magnitude.html. [Accessed 8 07 2020].

[10] H. Ha, "viblo.asia," [Online]. Available: https://viblo.asia/p/tim-hieu-ve-phuong-phap-mo-ta-dac-trung-hog-histogram-of-oriented-gradients-V3m5WAwxZO7. [Accessed 7 07 2020].

[11] pypi.org, "CV2," [Online]. Available: https://pypi.org/project/opencv-python/. [Accessed 15 7 2020].

[12] sklearn, "sklearn.feature," [Online]. Available: https://scikit-image.org/docs/dev/api/skimage.feature.html. [Accessed 15 07 2020].

[13] sklearn, "sklearn.metrics," [Online]. Available: https://scikit-learn.org/stable/modules/classes.html. [Accessed 15 07 2020].

[14] sklearn, "sklearn.neighbors," [Online]. Available: https://scikit-learn.org/stable/modules/classes.html. [Accessed 15 7 2020].

[15] sklearn, "sklearn.linear_model," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. [Accessed 15 07 2020].

[16] sklearn, "sklearn.svm," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed 15 07 2020].

[17] tensorflow, "tensorflow.keras.model," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/Model. [Accessed 15 07 2020].

[18] tensorflow, "tensorflow.keras.layers," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers. [Accessed 15 07

2020].