

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH



ĐỒ ÁN TỔNG HỢP AI

Báo cáo cuối kì - Học kì 241

*Separating and authenticating sign and stamp
in document text*

Giáo viên hướng dẫn: Trần Tuấn Anh
Mai Xuân Toàn
Trần Hồng Tài

Nhóm: 6

Sinh viên: Nguyễn Tất Đạt - 2210697
Nguyễn Minh Đạt - 2210693
Trần Đức Trí Cường - 2210443
Lê Đỗ Minh Anh - 2252023

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 12 - 2024



Họ và tên	MSSV	Công việc	Phần trăm hoàn thành
Nguyễn Tất Đạt	2210697		100%
Nguyễn Minh Đạt	2210693		100%
Trần Đức Trí Cường	2210443		100%
Lê Đỗ Minh Anh	2252023		100%



Contents

1 Giới thiệu sơ lược về đề tài	3
2 Cơ sở lý thuyết	3
2.1 Mạng thần kinh nhân tạo	3
2.1.1 Định nghĩa	3
2.1.2 Các lớp phổ biến trong Hidden Layer	4
2.1.3 Optimizer và Loss	7
2.2 Bài toán nhận diện vật thể (Object detection)	9
2.2.1 Mạng RCNN	9
2.2.2 Mạng Fast-RCNN	10
2.2.3 Mạng Faster-RCNN	11
2.2.4 YOLO	13
2.3 Bài toán phân loại vật thể (Classification)	14
2.3.1 Các giải thuật cơ bản Machine learning	14
2.3.2 Các mô hình Deeplearning	21
3 Dữ liệu được sử dụng trong đề tài	27
3.1 Dữ liệu stamp	27
3.2 Dữ liệu chữ ký	30
4 Mô hình và Kết quả các mô hình được sử dụng cơ bản	30
4.1 Nhận diện và phân loại stamp	30
4.2 Nhận diện và phân loại chữ ký	34
5 Tổng kết	34

1 Giới thiệu sơ lược về đề tài

- Ngày nay trong các tài liệu pháp lý như hợp đồng hoặc biên bản,... con dấu (stamp) và chữ ký đóng vai trò quan trọng, thể hiện tính xác thực và cam kết của các bên liên quan.
 1. Con stamp thường đại diện cho: xác thực tư cách pháp nhân, đảm bảo tính pháp lý, ngăn ngừa giả mạo.
 2. Chữ ký thường đại diện cho: xác nhận sự đồng ý, ràng buộc trách nhiệm, xác minh danh tính.
- Từ hai ý trên ta có thể thấy con stamp và chữ ký có vai trò rất quan trọng trong cuộc sống, và để thực hiện được tốt các vai trò đó cả con stamp và chữ ký đều phải thực hiện được tốt tính chất ngăn ngừa tính giả mạo và xác minh danh tính. Vì nếu không thực hiện được tính chất này thì nó có thể bị làm giả và không còn ý nghĩa gì nữa.
- Dù với thời trước nó đã hoàn thành khá ổn nhưng với ngày nay, công nghệ ngày càng phát triển, các con stamp và chữ ký giả gây khó khăn cho việc nhận biết chúng bằng mắt thường. Nên nhóm em đã nghiên cứu xây dựng lên một model AI để thực hiện điều đó thay cho con người.
- Đề tài nhóm em gồm 2 giai đoạn chính là:
 - **Giai đoạn 1:** Nhận biết con dấu và chữ ký đang nằm ở đâu trên tài liệu, từ đó trích xuất ra và đưa vào giai đoạn 2.
 - **Giai đoạn 2:** Sau khi đã trích xuất ra được con dấu và chữ ký, chúng em đưa cả 2 ảnh đó qua một mạng thần kinh nhân tạo để trích xuất ra các đặc trưng và quyết định xem là con dấu hoặc chữ ký đó có phải là chữ ký thật hay không.

2 Cơ sở lý thuyết

2.1 Mạng thần kinh nhân tạo

2.1.1 Định nghĩa

Mạng thần kinh nhân tạo (Artificial Neural Network) là một mô hình tính toán được lấy cảm hứng từ cấu trúc và chức năng của não người. Về cơ bản, mạng thần kinh nhân tạo là một tập hợp các đơn vị xử lý thông tin, hay còn gọi là các neuron, được kết nối với nhau bằng các trọng số. Các trọng số này cho phép mạng "học" từ dữ liệu thông qua một quá trình điều chỉnh lặp đi lặp lại.

Mạng thần kinh nhân tạo gồm ba loại lớp chính:

- Lớp đầu vào (Input Layer): là lớp đầu tiên trong mạng, nhận các đặc trưng của dữ liệu đầu vào và chuyển đến lớp tiếp theo. Số lượng neuron trong lớp đầu vào ứng với số lượng yếu tố đầu vào của bài toán mà ta cần giải quyết.
- Lớp ẩn (Hidden Layer): là các lớp trung gian trong mạng, các lớp này sẽ thực hiện các phép tính phức tạp để trích xuất ra được những đặc trưng quan trọng của dữ liệu. Một mạng có thể có nhiều lớp ẩn, và mỗi lớp có thể chứa hàng trăm hoặc hàng ngàn neuron. Các lớp ẩn thường sử dụng các hàm kích hoạt như ReLU (Rectified Linear Unit), Tanh (Hyperbolic Tangent) hoặc Sigmoid.
- Lớp đầu ra (Output Layer): là lớp cuối cùng trong mạng, cung cấp kết quả cuối cùng của dự đoán. Số lượng neuron trong lớp này sẽ phụ thuộc vào số lượng class trong bài toán phân loại.

2.1.2 Các lớp phô biến trong Hidden Layer

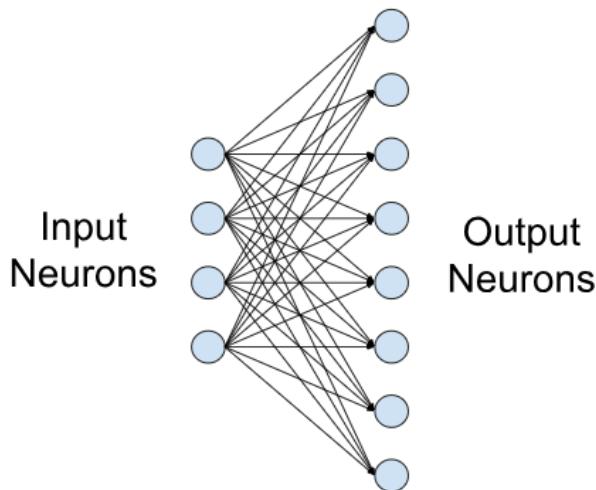
1. Fully Connected Layer

- Lớp kết nối đầy đủ, hay lớp dày, là loại lớp phô biến nhất trong mạng thần kinh nhân tạo. Mỗi neuron trong một lớp được kết nối với tất cả các neuron trong lớp kế tiếp, do vậy Fully connected layer thường được sử dụng ở đầu và cuối mạng, giúp tổng hợp và phân loại thông tin đã được xử lý qua các lớp khác.

$$y = W \cdot x + b \quad (1)$$

Trong đó:

- y là đầu ra của lớp.
- W là ma trận trọng số.
- x là vector đầu vào.
- b là vector bias.



Hình 1: Fully Connected Layer

2. Convolutional Layer

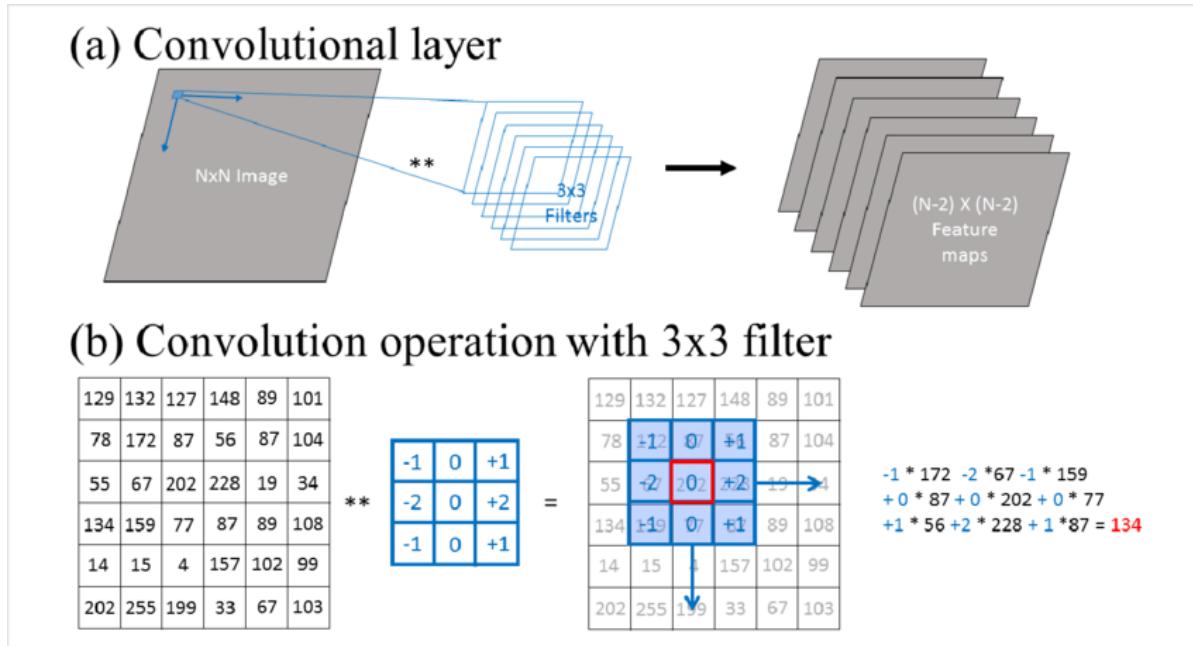
- Convolutional Layer thường được sử dụng trong mạng thần kinh tích chập (CNN) nhằm giải quyết các vấn đề về xử lý ảnh. Lớp này sẽ sử dụng các ma trận bộ lọc (filter) để trích xuất ra các đặc trưng không gian từ dữ liệu đầu vào. Mỗi bộ lọc sẽ quét qua dữ liệu và tính toán một tập hợp các feature maps.

$$y_{i,j}^k = \sum_{m,n} x_{i+m,j+n} \cdot w_{m,n}^k + b^k \quad (2)$$

Trong đó:

- $y_{i,j}^k$ là giá trị của ô lọc k tại vị trí (i, j) .
- $x_{i+m,j+n}$ là giá trị đầu vào tại vị trí $(i + m, j + n)$.

- $w_{m,n}^k$ là trọng số của ô lọc k tại vị trí (m, n) .
- b^k là giá trị bias của ô lọc k .



Hình 2: Convolutional Layer

3. Pooling Layer

- Pooling Layer giúp giảm số lượng đặc trưng từ ngoài vào (spatial dimensions) mà không làm mất đi các đặc trưng quan trọng đã trích xuất. Hai loại phổ biến là max pooling và average pooling:

- **Max pooling:** Chọn giá trị lớn nhất từ một ô lọc

$$y_{i,j} = \max(x_{i,j}^{(m,n)}) \quad (3)$$

- **Average pooling:** Tính giá trị trung bình của các giá trị trong ô lọc

$$y_{i,j} = \frac{1}{m \times n} \sum_{m,n} x_{i,j} \quad (4)$$

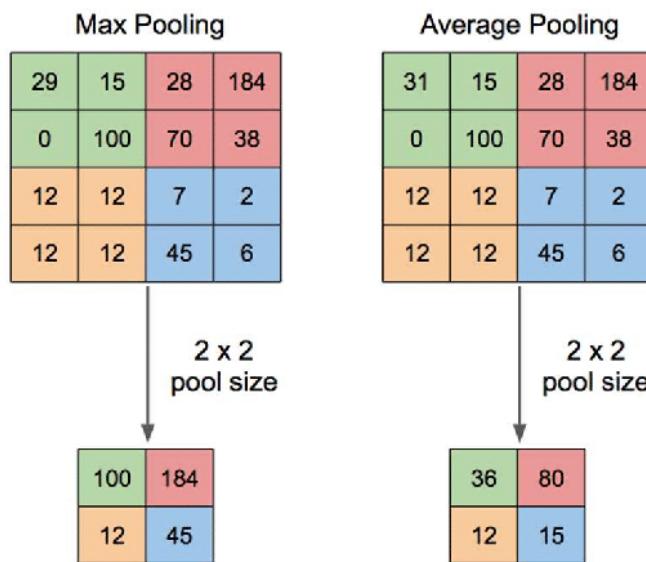
Trong đó:

* $y_{i,j}$ là đầu ra tại vị trí (i, j) .

* $x_{i,j}^{(m,n)}$ là đầu vào tại vị trí (i, j) trong vùng pooling $m \times n$.

4. Dropout Layer

- Để tránh hiện tượng overfitting, lớp Dropout sẽ ngẫu nhiên loại bỏ một số neuron trong quá trình học, cải thiện khả năng tổng quát hóa của mạng.

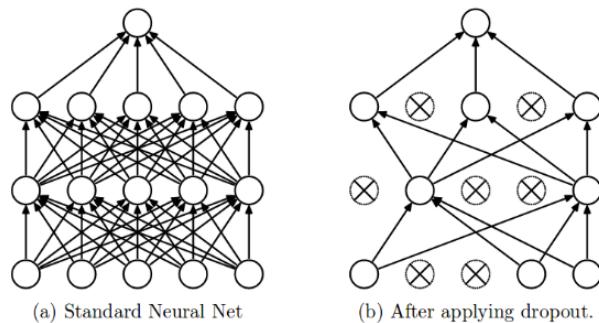


Hình 3: Pooling layer

$$y = \frac{1}{p} \cdot (x \cdot r) \quad (5)$$

Trong đó:

- y là đầu ra.
- p là xác suất giữ lại neuron.
- x là vector đầu vào.
- r là vector nhị phân ngẫu nhiên (0 hoặc 1) với xác suất giữ lại là p .



Hình 4: Dropout

5. Batch Normalization Layer

- Về cơ bản, BatchNorm Layer sẽ chuẩn hóa input của mỗi lớp bằng cách thay đổi giá trị trung

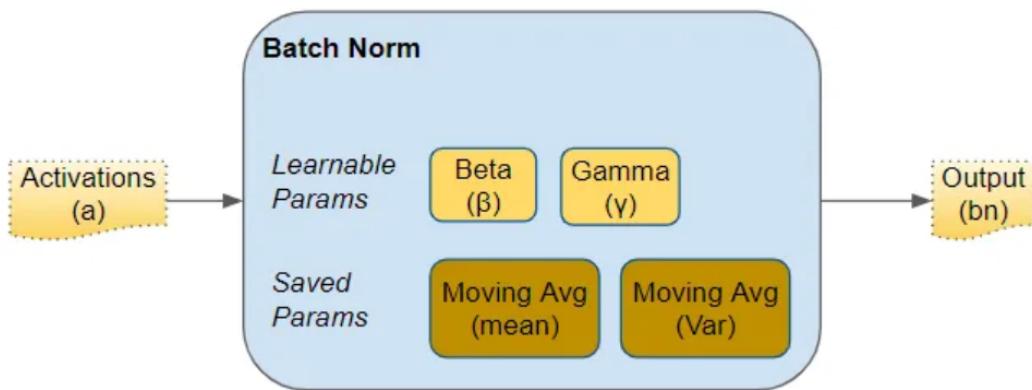
bình và phương sai để giúp cho mạng học nhanh và ổn định hơn.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{2(k)} + \varepsilon}} \quad (6)$$

$$y^{(k)} = \gamma^{(k)} \cdot \hat{x}^{(k)} + \beta^{(k)} \quad (7)$$

Trong đó:

- $\hat{x}^{(k)}$ là giá trị chuẩn hóa.
- $\mu_B^{(k)}$ là giá trị trung bình của batch B .
- $\sigma_B^{2(k)}$ là phương sai của batch B .
- ε là một số nhỏ để tránh chia cho 0.
- $\gamma^{(k)}$ và $\beta^{(k)}$ là các tham số học được.



Hình 5: BatchNorm Layer

2.1.3 Optimizer và Loss

1. **Optimizer:** là các thuật toán được sử dụng để điều chỉnh trọng số của mạng nhằm tối ưu quá trình học.
 - Stochastic Gradient Descent (SGD): là phương pháp cổ điển và phổ biến để tối ưu hóa mạng. Thay vì sử dụng toàn bộ tập dữ liệu cho mỗi bước cập nhật, SGD chỉ sử dụng một phần nhỏ của dữ liệu (mini-batch), nhờ đó giảm thời gian tính toán và tăng tốc quá trình học.
 - Adam (Adaptive Moment Estimation): là một phương pháp optimizer kế thừa những đặc tính tốt nhất của SGD và RMSprop. Adam tự động điều chỉnh tỉ lệ học cho từng trọng số thông qua việc tính toán động lượng của gradient, làm giảm độ trễ gradient.
 - RMSprop (Root Mean Square Propagation): RMSprop điều chỉnh tỉ lệ học của mỗi trọng số dựa trên độ lớn của gradient trước đó, tránh việc rơi vào các điểm cực trị không mong muốn.
2. **Loss Function:** giúp đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực tế. Hay nói cách khác, mục tiêu của bài toán là làm sao giảm thiểu giá trị của loss function. Hàm Loss được sử dụng



phổ biến là hàm MSE (Mean Squared Error):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (8)$$

3. **Cross-Entropy Loss:** giúp đo lường sự khác biệt giữa phân phối dự đoán và phân phối thực tế của các lớp. Cross-Entropy Loss thường được sử dụng trong các bài toán phân loại, đặc biệt hiệu quả khi sử dụng với hàm kích hoạt softmax.

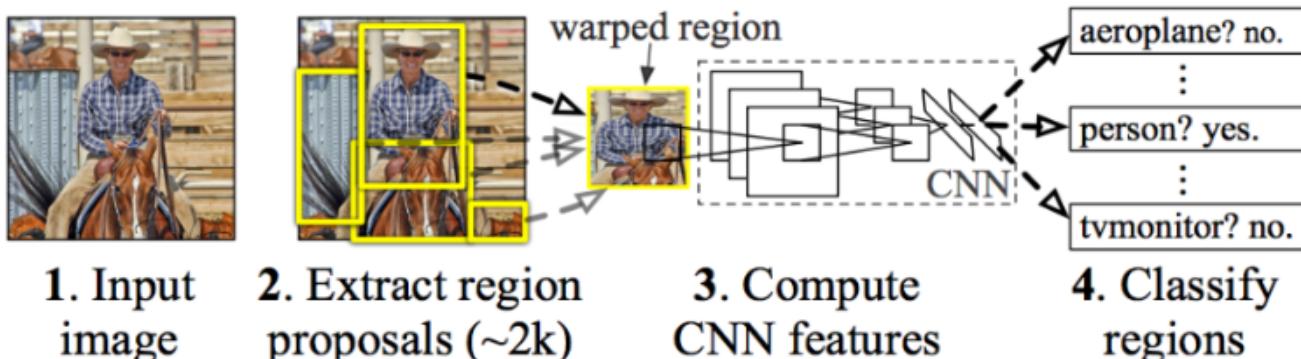
$$J = -\frac{1}{N} \sum_{i=1}^N y_i \log(\hat{y}_i) \quad (9)$$

2.2 Bài toán nhận diện vật thể (Object detection)

- Sau khi đã tìm hiểu được Neural Network là gì và các layer phổ biến, chúng ta sẽ đi đến bài toán phổ biến sử dụng mạng neural đầu tiên: Object detection.
- Object Detection (phát hiện đối tượng) là một nhánh quan trọng trong lĩnh vực Thị giác máy tính (Computer Vision), tập trung vào việc nhận diện và định vị các đối tượng trong một hình ảnh hoặc video. Công nghệ này không chỉ xác định đối tượng thuộc loại gì (classification) mà còn tìm ra vị trí chính xác của chúng thông qua các khung giới hạn (bounding box).
- Sau khi CNN cũng như các lớp layer ra đời, máy tính đã có thể trích xuất đặc trưng, nhận diện, phân loại vật thể có trong ảnh. Nhưng điểm yếu của phương pháp này là nó chỉ phân loại được ảnh chứa 1 vật thể và vật thể đó phải chiếm gần hết khung ảnh. Trong ảnh có nhiều vật thể, nó không thể xác định được những vật thể đang nằm ở đâu cũng như không thể phân biệt được các vật thể trong ảnh đó là những vật thể gì.

2.2.1 Mạng RCNN

- Để cải thiện điểm yếu của mạng MLP đơn giản trên, vào ngày 22/10/2014 các tác giả Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik đã cho ra đời phương pháp mạng RCNN.



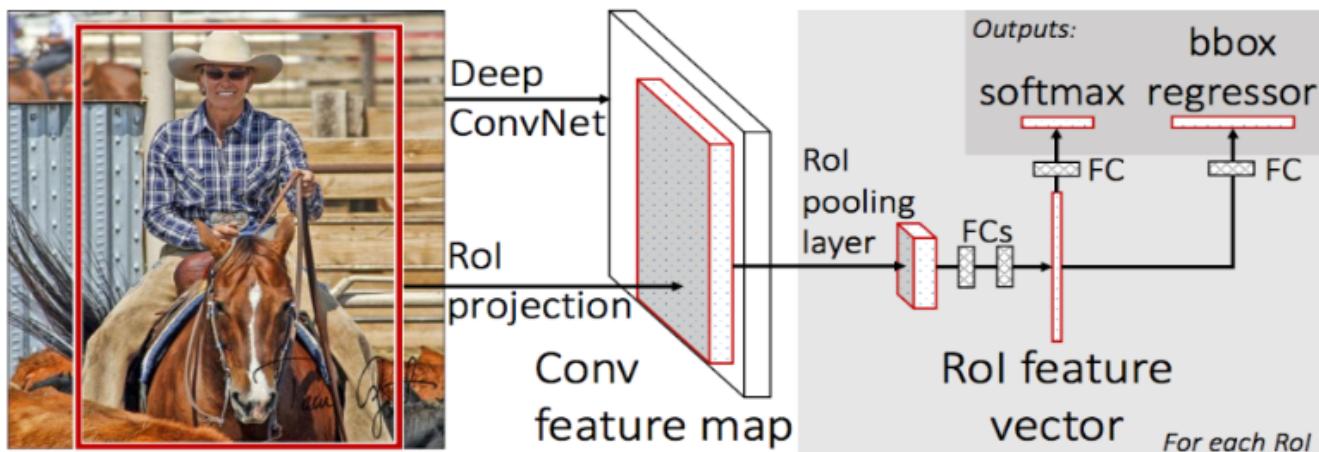
Hình 6: RCNN Pipeline

- Mạng RCNN gồm 3 bước chính:

- Đề xuất vùng (Region Proposals):** Mạng RCNN giải quyết vấn đề không thể nhận diện nhiều vật thể trên cùng một ảnh bằng giải pháp đưa ra một giải thuật đề xuất vùng đơn giản **Selection Search**. Giải thuật này tạo ra các vùng đề xuất trên mỗi hình ảnh (thường là 2000 vùng ảnh).
 - Trích xuất đặc trưng CNN (Feature Extraction):** Sau đó 2000 vùng ảnh được chuẩn đoán xem có thể chứa vật không, sau đó lần lượt được đưa qua mạng CNN để trích xuất các đặc trưng của các vùng ảnh đó.
 - Phân loại vùng (Region Classification):** Với từng đặc trưng lại được đi qua lớp SVM hoặc SOFTMAX để phân loại ảnh đó tương tự mạng CNN.
- Mô hình mạng này đã giải quyết được điểm yếu của CNN. Nhưng cũng có thể nhận thấy, ở trường hợp tệ nhất, với mỗi input ảnh ta phải xử lý 2000 phân vùng ảnh dự đoán của giải thuật **Selection Search**. Người ta ước tính với mỗi ảnh như vậy cần xấp xỉ 47s để xử lý.

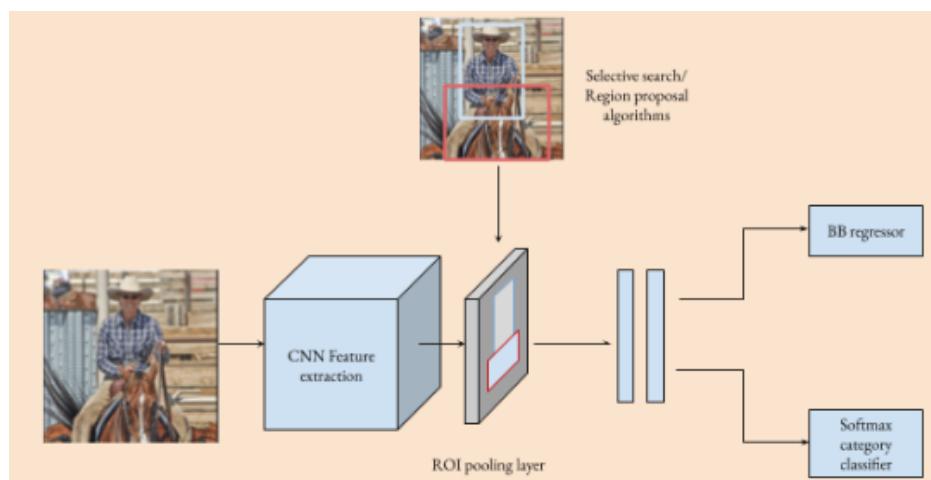
2.2.2 Mạng Fast-RCNN

- Các tác giả của RCNN đã tự cải thiện tốc độ của mô hình bằng cách giảm chi phí tính toán của máy tính, từ đó Fast-RCNN ra đời.



Hình 7: Fast-RCNN Architect

- Kiến trúc của mạng Fast-RCNN khá tương đồng với kiến trúc của mạng RCNN, gồm 4 bước chính:
 - Đè xuất các vùng ROI:** Mạng cũng sử dụng giải thuật **Selection Search** để đưa ra các vùng đề xuất, lúc này các vùng này còn được gọi là ROI.
 - Trích xuất đặc trưng:** Sau đó đưa **bức ảnh ban đầu** (input) qua một lớp CNN để trích xuất đặc trưng.
 - Ánh xạ ROI (ROI Pooling):** Sau khi đã trích xuất được đặc trưng trên vùng ảnh gốc, các ROI được xác định từ ban đầu sẽ được ánh xạ lên bản đồ đặc trưng vừa trích xuất được và chuẩn hóa kích thước bằng ROI Pooling.



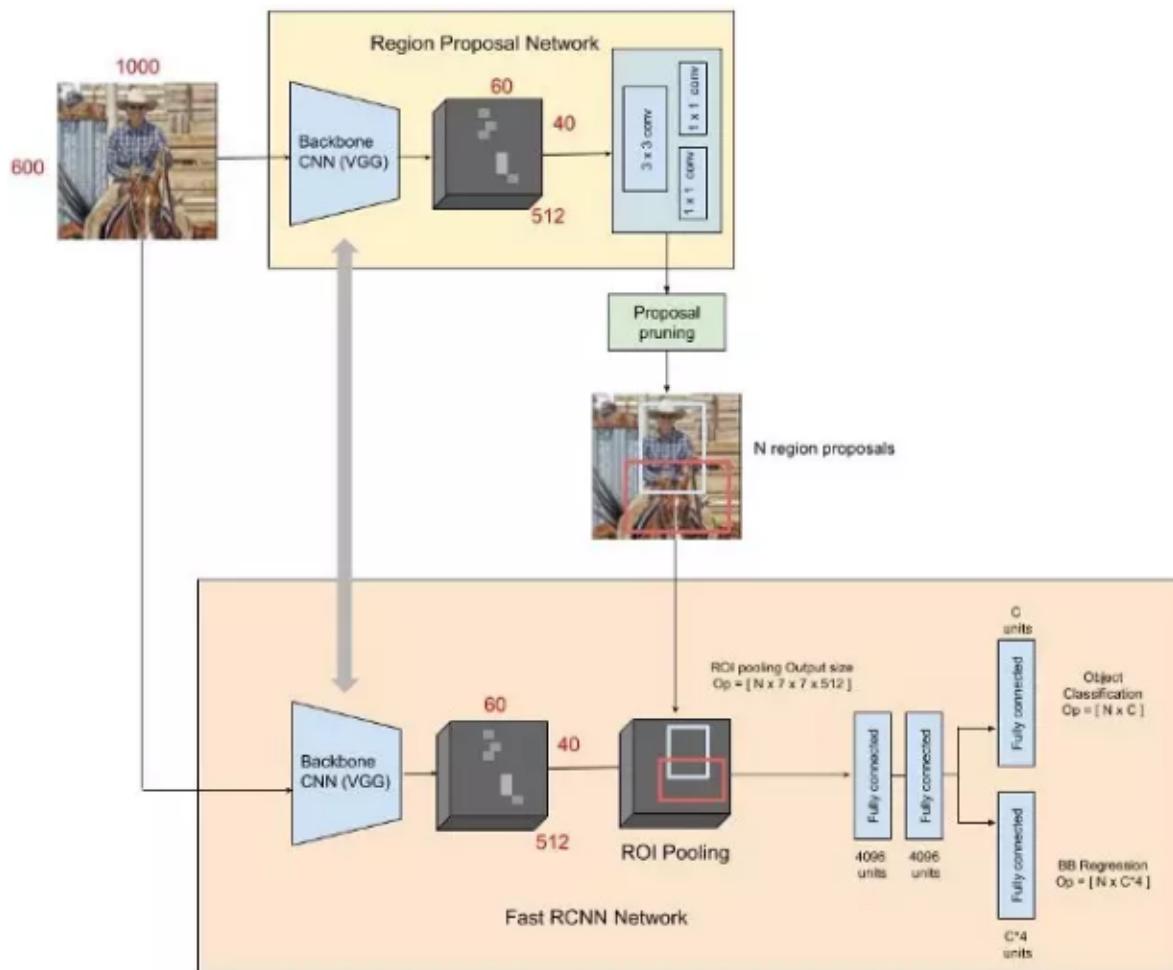
Hình 8: ROI Pooling example

- Phân loại:** Các đặc trưng từ ROI Pooling được đưa qua một lớp FULLY CONNECTED và SOFTMAX để phân loại và tinh chỉnh hộp giới hạn (Bounding Box).

- Nhìn chung mạng Fast-RCNN đã phần nào giảm được thời gian thực thi so với RCNN nhờ vào việc không đưa lần lượt 2000 ROI qua từng lớp CNN để trích xuất đặc trưng mà thực hiện trích xuất duy nhất 1 lần trên ảnh gốc kết hợp với cơ chế ánh xạ ROI Pooling giúp ánh xạ chính xác đặc trưng đó đang thuộc ROI nào. Từ đó giúp giảm hơn một nữa độ phức tạp tài nguyên và thời gian so với mạng RCNN.
- Nhưng hạn chế lớn nhất vẫn là giải thuật **Selection Search** vẫn còn quá lâu để trích xuất ra được 2000 vùng ROI. Giải thuật này như một thắt nút cổ chai về thời gian của mô hình tổng.

2.2.3 Mạng Faster-RCNN

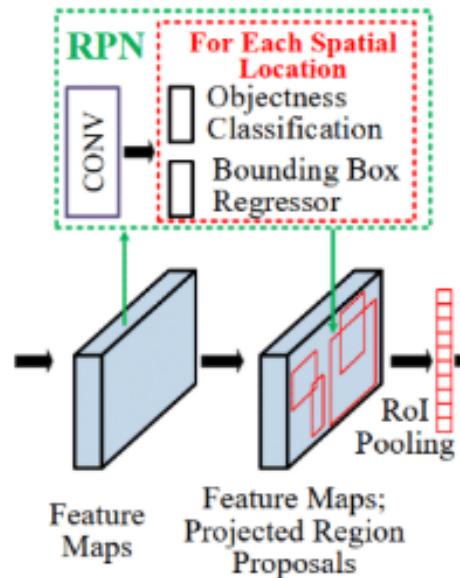
- Nhận thấy vấn đề nút thắt của giải thuật **Selection Search**, tác giả Shaoqing Ren đã cho ra mắt mô hình Faster-RCNN như một cải tiến cuối cùng và mạnh mẽ nhất của mô hình họ nhà RCNN này.



Hình 9: Faster-RCNN Architect

- Faster-RCNN là sự kết hợp bởi 2 modules: **Region Proposal Network (RPN)** với vai trò là đề xuất ra các vùng và **Fast-RCNN Network** với vai trò xử lý các vùng đã được đề suất, bao gồm 4 bước chính:

1. **Trích xuất đặc trưng:** Đầu tiên đưa ảnh qua CNN để trích xuất các đặc trưng của ảnh (thường dùng kiến trúc VGG-16 hoặc ResNet50).
2. **Region proposal network (RPN):** Tiếp đó các đặc trưng được xử lý qua lớp mạng RPN, mạng này đảm nhiệm nhiệm vụ của giải thuật Selection Search trước đó. Mạng này có 4 bước chính:
 - Đầu tiên nhận đầu vào feature map của bước trước và áp dụng một **cửa sổ trượt (Sliding Window)** lên feature map.
 - Sau đó tại mỗi vị trí của cửa sổ trượt chúng ta dự đoán đồng thời nhiều region proposal cùng một lúc, với k là số proposal tương ứng với mỗi vị trí. K proposals được tham chiếu hóa tới k boxes, còn được gọi là anchor.
 - Tiếp đó xác định mỗi anchor là đối tượng hay chỉ là background.
 - Cuối cùng tinh chỉnh tọa độ các anchors qua các lần train để dự đoán chính xác hơn các vùng đề xuất. Một tập hợp các vùng đề xuất với xác xuất có chứa đối tượng đã được tạo ra trên feature map.

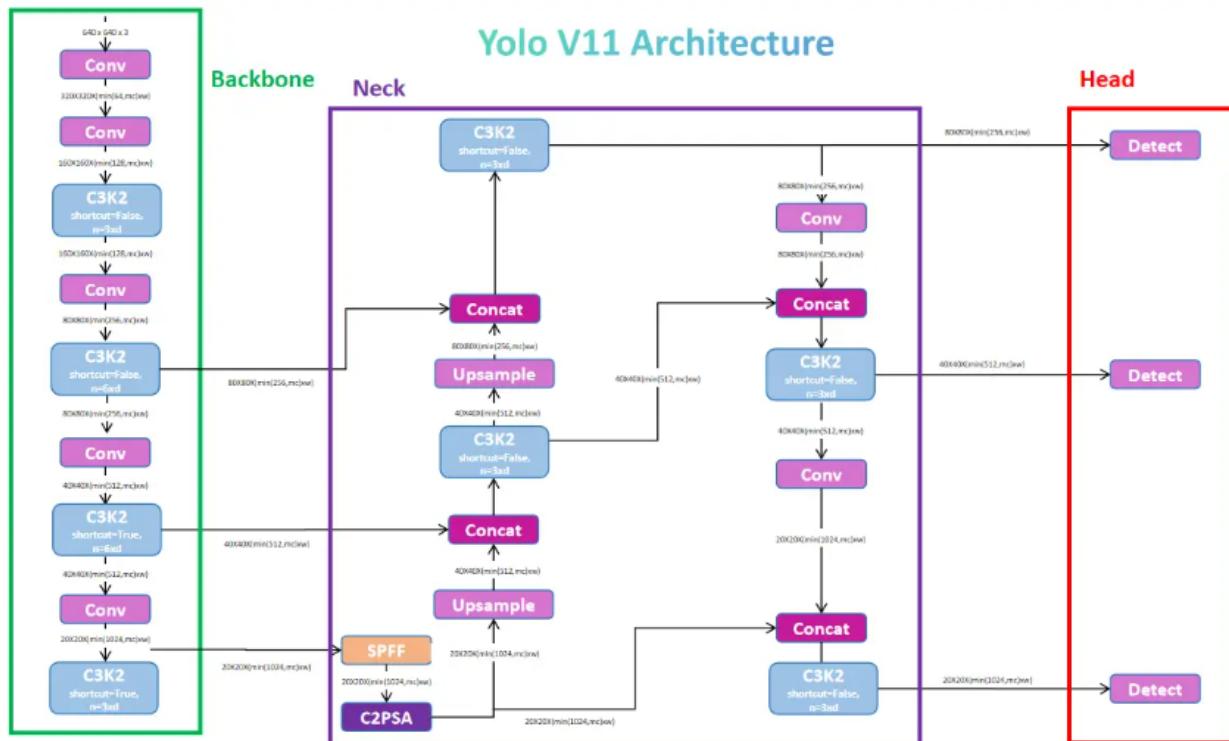


Hình 10: RPN Architect

3. **ROI Pooling:** Tuy nhiên các vùng đề xuất có thể không cùng shape nên phải cho qua một lớp ROI Pooling để ánh xạ chính xác các vùng đó về cùng size trên feature map.
4. **Phân loại:** Cuối cùng tương tự mạng Fast-RCNN, các đặc trưng từ ROI Pooling được đưa qua một lớp FULLY CONNECTED và SOFTMAX để phân loại và tinh chỉnh hộp giới hạn (Bounding Box).
 - Có thể thấy nhờ vào việc thay thế giải thuật Selection Search truyền thống bằng một mạng RPN được train để dự đoán được các vùng, kết quả cho ra tốt hơn rất nhiều và cũng đã gỡ được nút thắt cổ chai của mô hình Fast-RCNN.

2.2.4 YOLO

- Vào năm 2016 Dù mô hình Faster-RCNN đã thực hiện khá tốt nhiệm vụ Object detection nhưng YOLO (You Only Look Once) ra đời đã cải thiện khủng khiếp độ chính xác và tốc độ detect của mô hình nhờ vào kiến trúc cực kì phức tạp của nó.
- Đúng với cái tên YOLO, nghĩa là mạng chỉ cần xem qua hình ảnh một lần duy nhất để dự đoán cả vị trí và danh mục của các đối tượng trong ảnh. Điều này khác biệt so với các phương pháp trước đó như R-CNN, Faster R-CNN, vốn yêu cầu nhiều bước tính toán phức tạp và tốn thời gian.
- YOLO hoạt động dựa trên 3 bước chính:
 - Chia hình thành SxS ô lưới.
 - Mỗi ô dự đoán B bounding box, cùng với confidence score (điểm tin cậy) thể hiện mức độ chắc chắn rằng box chứa đối tượng.
 - Kết hợp các dự đoán để xuất ra bounding box cuối cùng cùng với danh mục đối tượng.
- YOLO hiện tại đã có tới phiên bản v11, với kiến trúc rất phức tạp nên nhóm chỉ ứng dụng cho bài toán chứ không bàn luận nhiều.



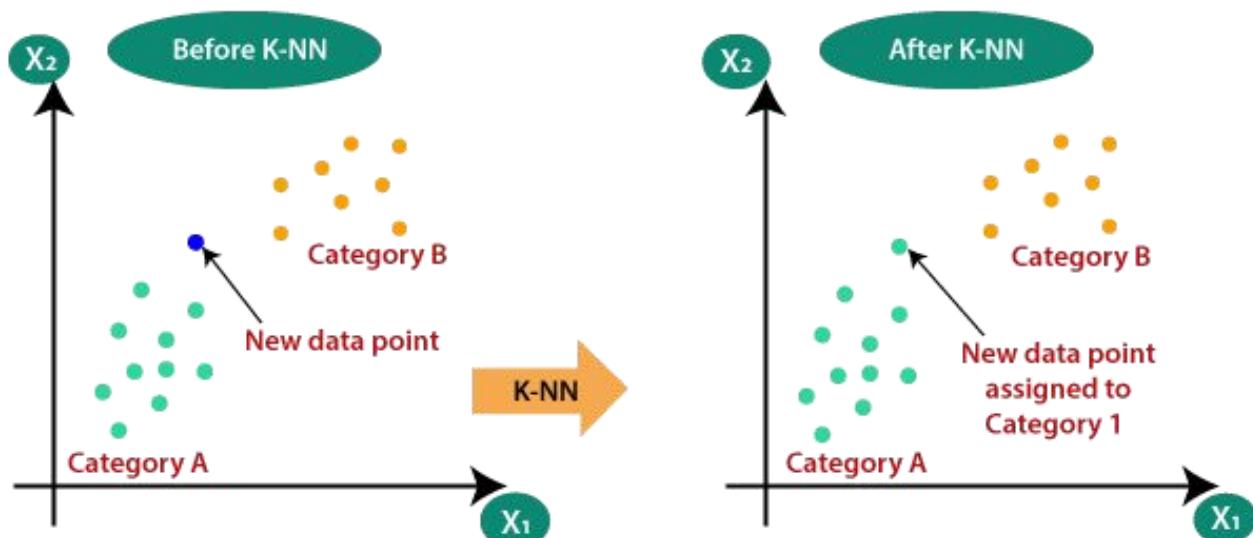
Hình 11: YOLOv11 Architect

2.3 Bài toán phân loại vật thể (Classification)

- Classification (phân loại) là một phương pháp trong học máy (Machine Learning) và trí tuệ nhân tạo (Artificial Intelligence) dùng để dự đoán và gán nhãn (label) cho các đối tượng dữ liệu vào các nhóm hoặc lớp (classes) đã xác định trước. Kỹ thuật này thường được áp dụng trong các bài toán mà đầu ra (output) thuộc về một tập hợp hữu hạn các giá trị rời rạc.
- Mục tiêu của classification là học từ một tập dữ liệu huấn luyện (training dataset), nơi mỗi mẫu dữ liệu đã được gán nhãn, để xây dựng một mô hình dự đoán (predictive model). Mô hình này sau đó sẽ được sử dụng để phân loại các mẫu dữ liệu mới, chưa biết nhãn.

2.3.1 Các giải thuật cơ bản Machine learning

- KNN



Hình 12: Mô hình K-Nearest-Neighbour

Cơ sở lý thuyết

K-Nearest Neighbors (KNN) là một thuật toán học máy không tham số, thường được sử dụng cho các bài toán phân loại và hồi quy. Nguyên tắc cơ bản của KNN là dựa vào khoảng cách giữa các điểm dữ liệu để xác định nhãn của một điểm mới dựa trên nhãn của k điểm lân cận nhất.

1. Cách hoạt động

- Phân loại:** Một mẫu mới sẽ được gán nhãn dựa trên nhãn của k điểm gần nhất. Thông thường, nhãn được chọn là nhãn chiếm đa số trong k điểm lân cận.
- Hồi quy:** Giá trị của mẫu mới được tính bằng trung bình trọng số (hoặc trung bình cộng) của các giá trị k điểm lân cận.

2. Công thức tính khoảng cách

Phổ biến nhất trong KNN là sử dụng khoảng cách Euclid, được định nghĩa như sau:

$$d(x, x') = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}$$

Trong đó:

- x, x' : Hai điểm trong không gian n chiều.
- x_i, x'_i : Giá trị của thuộc tính thứ i của hai điểm.

Ngoài khoảng cách Euclid, một số cách tính khoảng cách khác cũng thường được sử dụng:

- **Khoảng cách Manhattan:**

$$d(x, x') = \sum_{i=1}^n |x_i - x'_i|$$

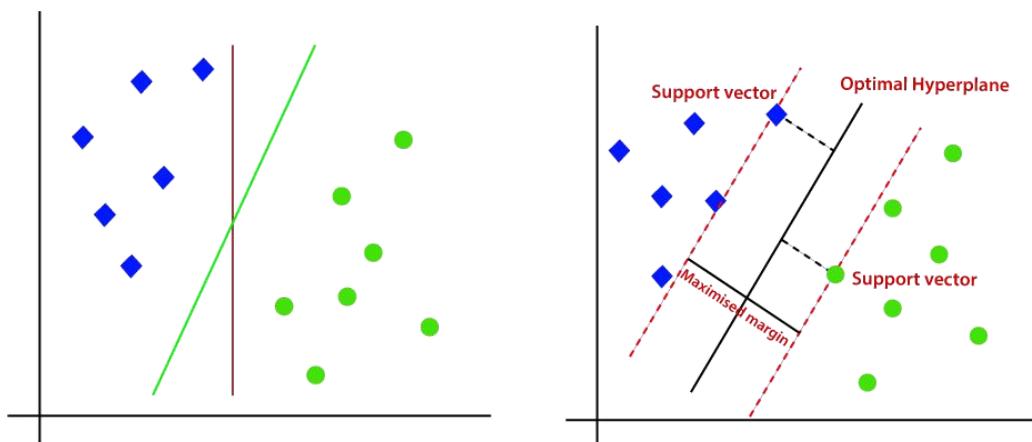
- **Khoảng cách Minkowski:**

$$d(x, x') = \left(\sum_{i=1}^n |x_i - x'_i|^p \right)^{1/p}$$

3. Các yếu tố ảnh hưởng đến hiệu quả của KNN

- **Giá trị của k :** Giá trị k quyết định số lượng điểm lân cận được xem xét. Giá trị k nhỏ có thể dẫn đến overfitting, trong khi giá trị k lớn có thể làm mất đi tính chi tiết của dữ liệu.
- **Độ đo khoảng cách:** Lựa chọn độ đo khoảng cách phù hợp sẽ ảnh hưởng lớn đến kết quả.
- **Chuẩn hóa dữ liệu:** Để tránh ảnh hưởng của các thuộc tính có đơn vị đo khác nhau, dữ liệu nên được chuẩn hóa trước khi áp dụng KNN.

- **SVM**



Hình 13: Mô hình Support Vector Machine

Cơ sở lý thuyết về mô hình SVM

Support Vector Machine (SVM) là một thuật toán học máy giám sát, được sử dụng phổ biến trong các bài toán phân loại và hồi quy. SVM hoạt động dựa trên nguyên tắc tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách dữ liệu thành các lớp khác nhau với khoảng cách lớn nhất.

1. Nguyên lý hoạt động

SVM tìm kiếm siêu phẳng tối ưu có dạng:

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

Trong đó:

- \mathbf{w} : Vector trọng số (weight vector).
- \mathbf{x} : Điểm dữ liệu (input vector).
- b : Sai số (bias).

Siêu phẳng tối ưu là siêu phẳng mà khoảng cách từ nó đến các điểm dữ liệu gần nhất (các support vector) được tối đa hóa. Khoảng cách này được gọi là **margin**.

2. Hàm mục tiêu của SVM

SVM tối ưu hóa bài toán sau:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

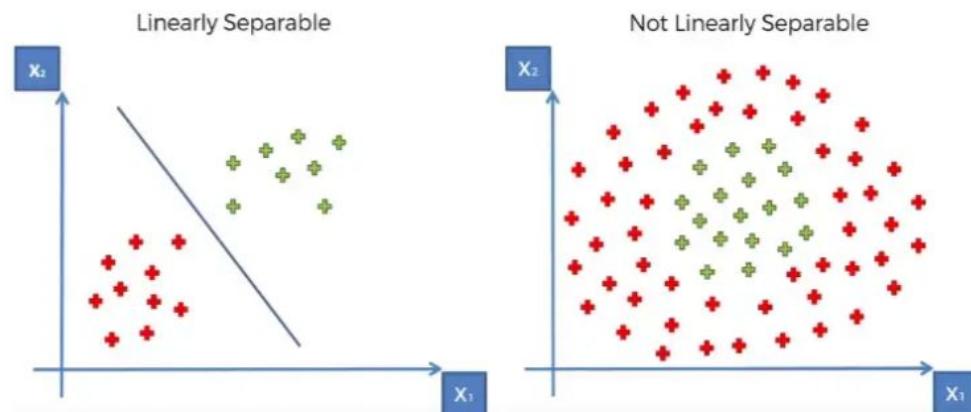
Với ràng buộc:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

Trong đó:

- y_i : Nhãn của điểm dữ liệu \mathbf{x}_i , $y_i \in \{-1, 1\}$.
- \mathbf{x}_i : Dữ liệu huấn luyện.

3. SVM với dữ liệu phi tuyến tính



Hình 14: Dữ liệu phi tuyến

Khi dữ liệu không thể phân tách tuyến tính, SVM sử dụng **hàm kernel** để ánh xạ dữ liệu sang không gian cao hơn, trong đó có thể tìm được siêu phẳng phân tách tuyến tính. Một số hàm kernel phổ biến:

- Kernel tuyến tính:

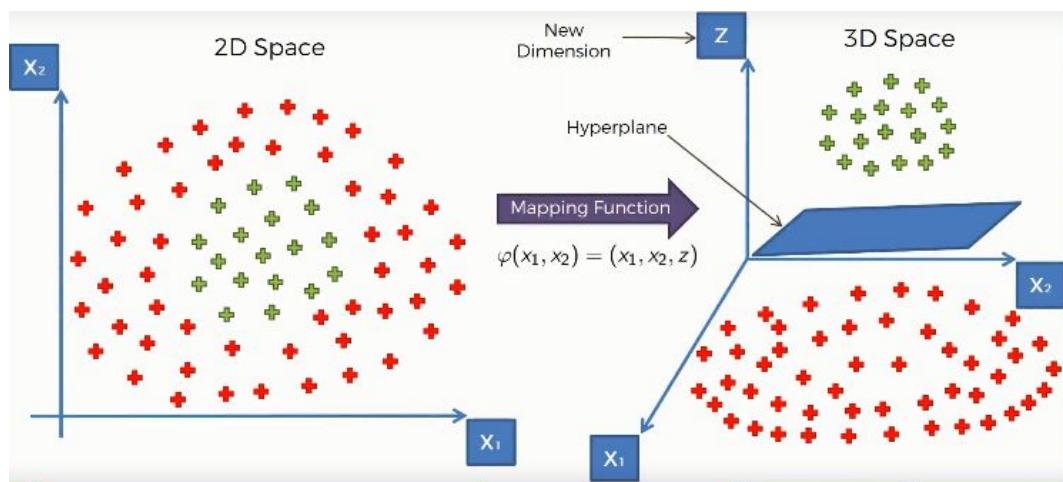
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$$

- Kernel Gaussian (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

- Kernel đa thức:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$$



Hình 15: Sử dụng Kernel để ánh xạ dữ liệu sang không gian cao hơn

4. SVM với dữ liệu không tách biệt hoàn toàn

Khi dữ liệu không thể tách biệt hoàn toàn, SVM cho phép một số điểm vi phạm margin thông qua biến bù ξ_i . Bài toán tối ưu trở thành:

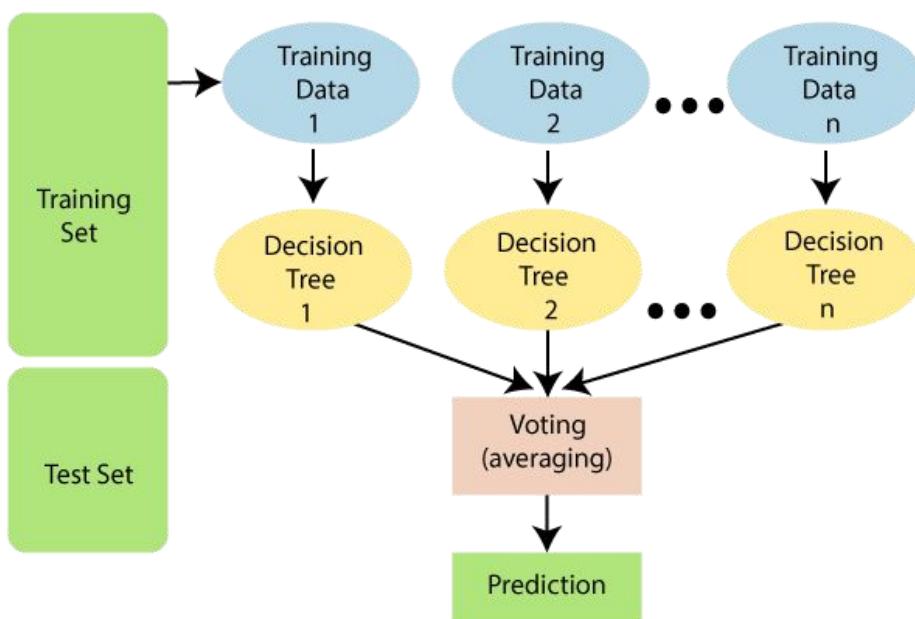
$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

Với ràng buộc:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

Trong đó C là siêu tham số điều chỉnh mức độ chấp nhận vi phạm margin.

- Random Forest



Hình 16: Mô hình Random Forest

Cơ sở lý thuyết

Random Forest là một thuật toán học máy thuộc nhóm phương pháp Ensemble Learning, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Thuật toán này hoạt động bằng cách xây dựng nhiều cây quyết định (Decision Trees) độc lập và tổng hợp kết quả để đưa ra dự đoán chính xác hơn.

1. Nguyên lý hoạt động

Random Forest dựa trên phương pháp **Bagging** (Bootstrap Aggregating), trong đó:

- Nhiều tập dữ liệu con được tạo ra từ tập dữ liệu gốc bằng cách chọn mẫu ngẫu nhiên có hoán lại (Bootstrap Sampling).
- Trên mỗi tập dữ liệu con, một cây quyết định (Decision Tree) được xây dựng.
- Kết quả cuối cùng được tổng hợp bằng cách:
 - * **Phân loại (Classification)**: Sử dụng bỏ phiếu đa số (majority voting) từ các cây.
 - * **Hồi quy (Regression)**: Sử dụng giá trị trung bình của các dự đoán từ các cây.

2. Công thức tổng hợp dự đoán

Đối với phân loại:

$$\hat{y} = \text{mode}\{h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_m(\mathbf{x})\}$$

Đối với hồi quy:

$$\hat{y} = \frac{1}{m} \sum_{i=1}^m h_i(\mathbf{x})$$

Trong đó:

- $h_i(\mathbf{x})$: Dự đoán từ cây i .
- m : Tổng số cây trong rừng.

3. Tính tầm quan trọng của đặc trưng

Tầm quan trọng của một đặc trưng được đánh giá dựa trên mức giảm impurity (độ nhiễu) tại mỗi nút cây, thường tính bằng Gini Index hoặc Entropy.

Gini Index:

$$G = 1 - \sum_{i=1}^C p_i^2$$

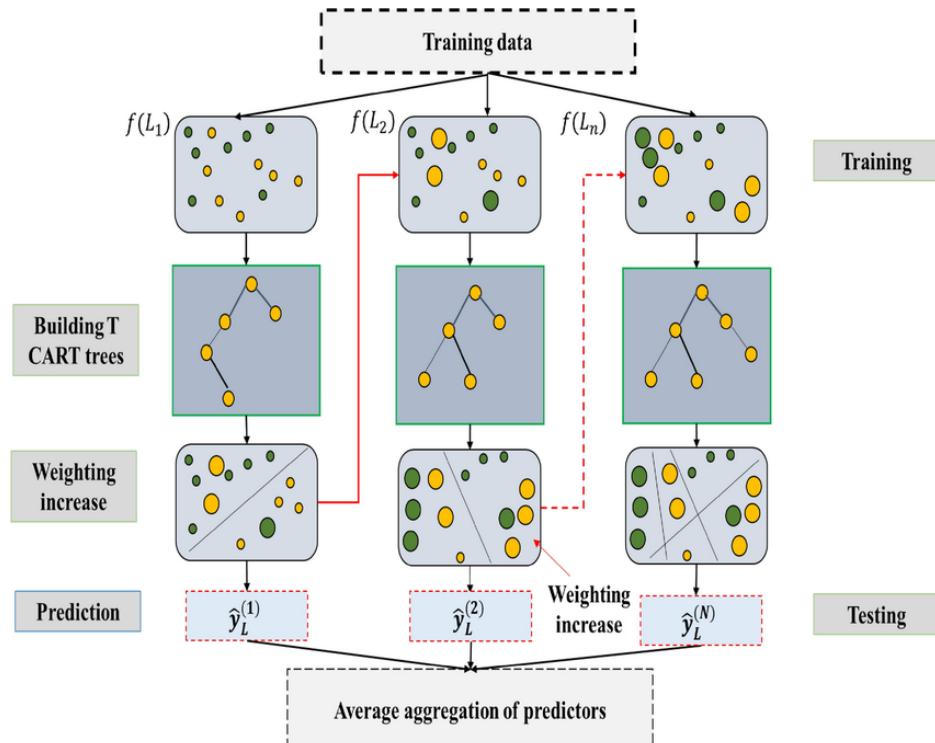
Entropy:

$$H = - \sum_{i=1}^C p_i \log_2 p_i$$

Trong đó:

- C : Số lượng lớp.
- p_i : Tỷ lệ phần tử thuộc lớp i tại nút đó.

- **XGBoost**



Hình 17: Boosting with Classification and Regression Trees



Cơ sở lý thuyết về mô hình XGBoost

XGBoost (Extreme Gradient Boosting) là một thuật toán học máy mạnh mẽ, thuộc nhóm phương pháp **Boosting**. Thuật toán này được thiết kế để tối ưu hóa cả hiệu suất dự đoán lẫn tốc độ xử lý, và thường được sử dụng trong các cuộc thi học máy cũng như các ứng dụng thực tế.

1. Nguyên lý hoạt động

XGBoost dựa trên ý tưởng của thuật toán **Gradient Boosting**, trong đó:

- Các mô hình cơ sở (Base Learners), thường là cây quyết định (Decision Trees), được xây dựng tuần tự.
- Mỗi mô hình mới được huấn luyện để sửa lỗi của mô hình trước đó, thông qua việc giảm thiểu một hàm mất mát (Loss Function) bằng phương pháp tối ưu gradient.

Công thức tổng hợp dự đoán:

$$\hat{y}_i = \sum_{k=1}^K f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}$$

Trong đó:

- \hat{y}_i : Giá trị dự đoán cho mẫu i .
- f_k : Hàm dự đoán của cây quyết định thứ k .
- \mathcal{F} : Tập hợp các cây quyết định.

2. Hàm mục tiêu

Hàm mục tiêu của XGBoost bao gồm hai thành phần:

$$\mathcal{L}(\Theta) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

Trong đó:

- $l(y_i, \hat{y}_i)$: Hàm mất mát đo lường sự khác biệt giữa giá trị thực y_i và dự đoán \hat{y}_i .
- $\Omega(f_k)$: Thành phần phạt để kiểm soát độ phức tạp của mô hình, được định nghĩa như sau:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|\mathbf{w}\|^2$$

- * T : Số lá của cây quyết định.
- * \mathbf{w} : Trọng số tại các lá.
- * γ và λ : Các siêu tham số để điều chỉnh mức độ phạt.

3. Tối ưu hóa hàm mục tiêu

Hàm mục tiêu được xấp xỉ bằng khai triển Taylor bậc hai:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t)$$

Trong đó:

- $g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$: Gradient của hàm mất mát.
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)2}}$: Đạo hàm bậc hai (Hessian) của hàm mất mát.

4. Chia nhỏ không gian

Khi xây dựng cây quyết định, XGBoost sử dụng công thức để tính điểm chất lượng của việc chia tách:

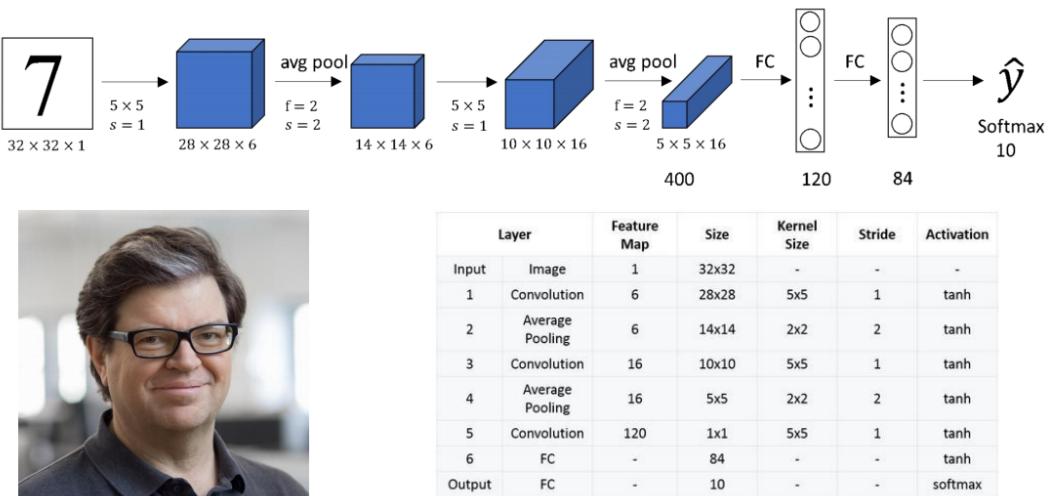
$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

Trong đó:

- G_L, G_R : Tổng gradient ở nhánh trái và phải.
- H_L, H_R : Tổng Hessian ở nhánh trái và phải.
- λ, γ : Siêu tham số phạt.

2.3.2 Các mô hình Deep learning

- LeNet (1998)



Hình 18: Mô hình LeNet (1998)

Cơ sở lý thuyết

LeNet là một trong những mạng nơ-ron tích chập (Convolutional Neural Network - CNN) đầu tiên, được Yann LeCun và cộng sự phát triển vào năm 1998. Mô hình này được thiết kế để xử lý bài toán nhận dạng chữ viết tay và số viết tay trên các hình ảnh grayscale kích thước nhỏ.

1. Tổng quan

LeNet đánh dấu sự khởi đầu của việc sử dụng các mạng CNN trong học sâu, với cấu trúc bao gồm các lớp tích chập (Convolutional Layers), lớp gộp (Pooling Layers), và lớp kết nối đầy đủ (Fully Connected Layers).

2. Kiến trúc mô hình

Kiến trúc của LeNet bao gồm 7 lớp (không tính lớp đầu vào), với thứ tự các lớp như sau:

- **Lớp tích chập 1:**

$$O = \frac{(I - F + 2P)}{S} + 1$$

Dầu vào: 32×32 , bộ lọc 5×5 , dầu ra 28×28 .

- **Lớp gộp 1:** Dầu vào 28×28 , dầu ra 14×14 .

- **Lớp tích chập 2:** Dầu vào 14×14 , bộ lọc 5×5 , dầu ra 10×10 .

- **Lớp gộp 2:** Dầu vào 10×10 , dầu ra 5×5 .

- **Lớp Fully Connected 1:** Dầu vào $5 \times 5 = 25$, dầu ra 120.

- **Lớp Fully Connected 2:** Dầu vào 120, dầu ra 84.

- **Lớp đầu ra:** Dầu vào 84, dầu ra 10 lớp.

3. Hàm kích hoạt

LeNet sử dụng các hàm kích hoạt sigmoid hoặc tanh:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

4. Quá trình huấn luyện

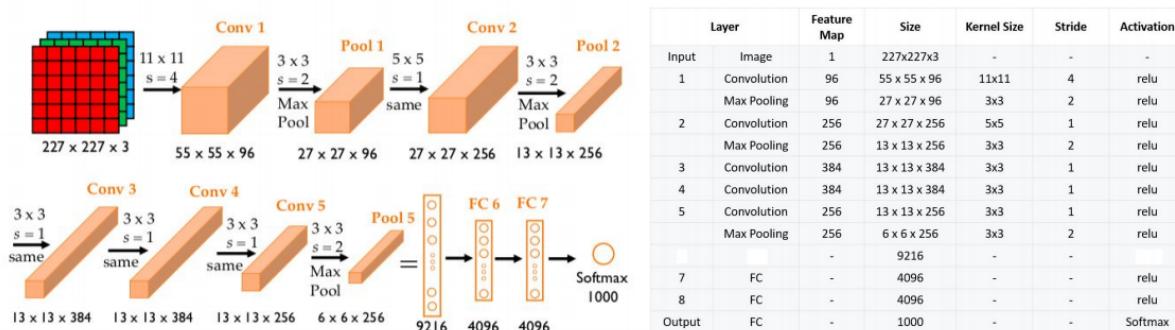
Hàm mất mát sử dụng:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

Trong đó:

- $y_{i,c}$: Nhãn thực của lớp c cho mẫu i .
- $\hat{y}_{i,c}$: Xác suất dự đoán của lớp c cho mẫu i .

- **AlexNet (2012)**



Hình 19: Mô hình AlexNet (2012)

Cơ sở lý thuyết

1. Tổng quan

AlexNet là một mạng nơ-ron tích chập (Convolutional Neural Network - CNN) được phát triển bởi Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton. Nó giành chiến thắng tại cuộc thi *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* năm 2012 với sai số top-5 chỉ 15.3%.

AlexNet mở rộng ý tưởng từ LeNet bằng cách xây dựng một mạng lớn hơn, sâu hơn và mạnh mẽ hơn, được hỗ trợ bởi các GPU để huấn luyện hiệu quả.

2. Kiến trúc mô hình

AlexNet bao gồm 8 lớp học tham số: 5 lớp tích chập (Convolutional Layers) và 3 lớp kết nối đầy đủ (Fully Connected Layers). Ngoài ra, mô hình sử dụng các kỹ thuật như ReLU, Dropout, Normalization và MaxPooling.

– Lớp tích chập:

- * Lớp đầu tiên: 96 filter 11×11 , stride bằng 4, đầu ra $55 \times 55 \times 96$.
- * Các lớp sau: Kích thước filter giảm dần ($5 \times 5, 3 \times 3$), số lượng filter tăng (256, 384, 384, 256).

– Lớp gộp: MaxPooling với kích thước 3×3 và stride bằng 2.

– Hàm kích hoạt: ReLU ($f(x) = \max(0, x)$).

– Lớp Fully Connected:

- * Hai lớp ẩn đầu tiên: Mỗi lớp 4096 nơ-ron.
- * Lớp đầu ra: Softmax với 1000 lớp:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

- **Dropout:** Loại bỏ ngẫu nhiên một số nơ-ron để giảm overfitting.
- **Local Response Normalization (LRN):** Chuẩn hóa cục bộ giúp tăng tính ổn định của việc kích hoạt.

3. Quá trình huấn luyện

AlexNet sử dụng thuật toán lan truyền ngược (Backpropagation) và tối ưu hóa bằng *Stochastic Gradient Descent (SGD)*.

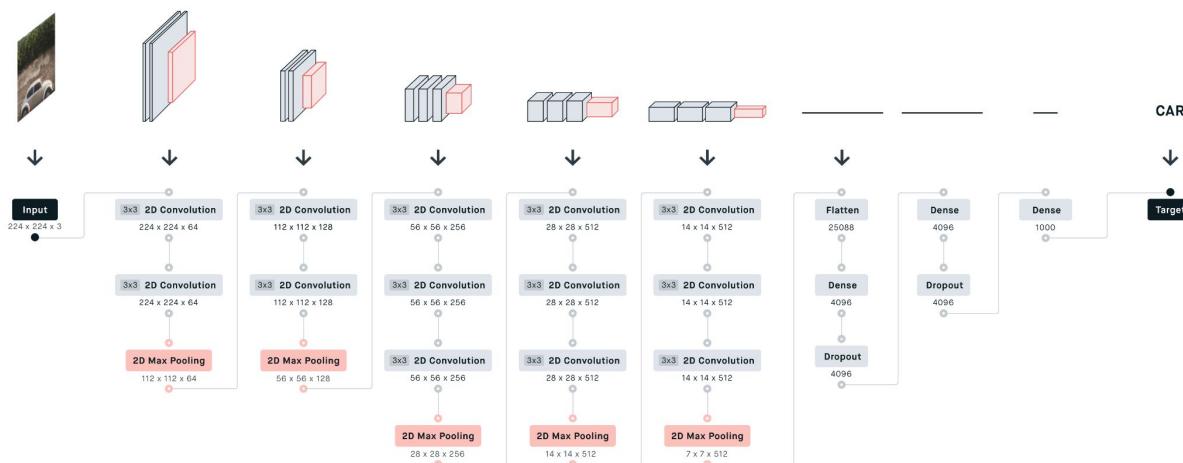
Hàm mất mát: Cross-Entropy Loss

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

Trong đó:

- $y_{i,c}$: Nhãn thực của lớp c cho mẫu i .
- $\hat{y}_{i,c}$: Xác suất dự đoán của lớp c cho mẫu i .

- **VGGNet (2014)**



Hình 20: Mô hình VGG (2014)

Cơ sở lý thuyết về VGGNet (2014)

1. Tổng quan

VGGNet là một mạng nơ-ron tích chập (CNN) được phát triển bởi nhóm nghiên cứu tại Đại học Oxford. Nó được thiết kế với ý tưởng sử dụng các kernel nhỏ (3×3) và tập trung vào việc tăng độ sâu của mạng để cải thiện hiệu năng.

2. Kiến trúc mô hình

VGGNet có nhiều phiên bản, trong đó VGG16 (16 lớp) và VGG19 (19 lớp) là phổ biến nhất.

- **Kernel kích thước nhỏ (3×3):** Giữ nguyên kích thước không gian nhưng giảm số lượng tham số.
- **Tăng độ sâu:** Mạng đạt hiệu năng cao bằng cách xếp chồng nhiều lớp tích chập.
- **Lớp gộp:** MaxPooling (2×2 , stride 2) giúp giảm kích thước không gian.
- **Lớp Fully Connected:** Ba lớp fully connected với số nơ-ron lần lượt là 4096, 4096, và 1000.
- **Hàm kích hoạt:** Sử dụng ReLU:

$$f(x) = \max(0, x)$$

- **Hàm mất mát:** Cross-Entropy Loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

3. Quá trình huấn luyện

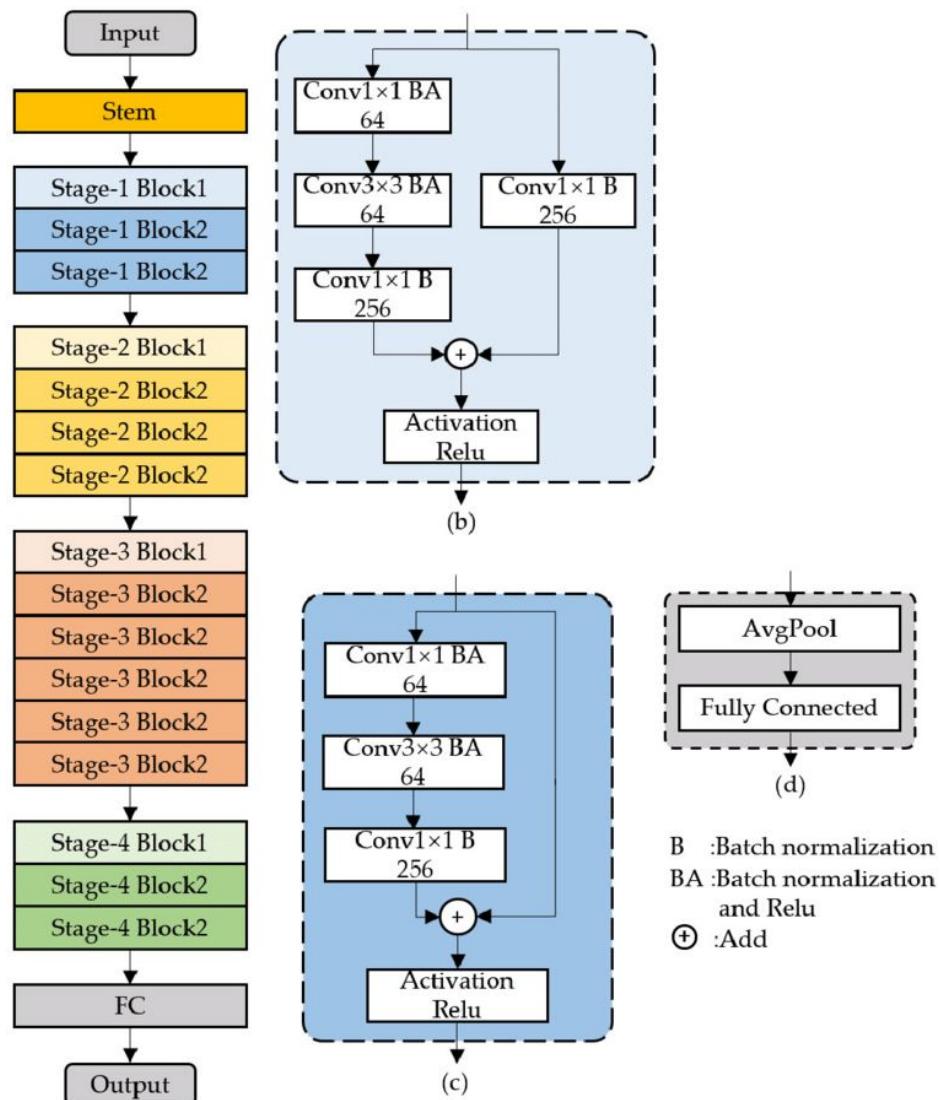
VGGNet được huấn luyện với *Stochastic Gradient Descent (SGD)*.

Dropout: Áp dụng ở các lớp fully connected với tỷ lệ 0.5.

Kiến trúc chi tiết

- **Input:** Hình ảnh kích thước $224 \times 224 \times 3$.
- **Các khối tích chập:**
 - * Block 1: 2 lớp tích chập 3×3 , 64 filter, MaxPooling.
 - * Block 2: 2 lớp tích chập 3×3 , 128 filter, MaxPooling.
 - * Block 3: 3 lớp tích chập 3×3 , 256 filter, MaxPooling.
 - * Block 4: 3 lớp tích chập 3×3 , 512 filter, MaxPooling.
 - * Block 5: 3 lớp tích chập 3×3 , 512 filter, MaxPooling.
- **Fully Connected Layers:** 3 lớp fully connected (4096, 4096, 1000).
- **Output:** Softmax với 1000 lớp.

- ResNet (2015)



Hình 21: Mô hình ResNet (2015)

Cơ sở lý thuyết

1. Tổng quan

ResNet (Residual Network) được giới thiệu bởi He et al. (2015) để giải quyết vấn đề "degradation problem". Điểm nhấn chính của ResNet là sử dụng *Residual Connections*, giúp thông tin và gradient được truyền qua mạng một cách hiệu quả.

2. Kiến trúc mô hình

ResNet được xây dựng dựa trên **Residual Block**:

– **Residual Block:**

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

Trong đó, $\mathcal{F}(\mathbf{x}, \{W_i\})$ là đầu ra của các lớp tích chập, và \mathbf{x} là đầu vào ban đầu.

– **Skip Connection:**

* *Identity Mapping:* Giữ nguyên đầu vào (\mathbf{x}).

* *Projection Shortcut:* Sử dụng tích chập 1×1 để điều chỉnh số kênh.

– **Hàm kích hoạt:** Sử dụng ReLU:

$$f(x) = \max(0, x)$$

3. Kiến trúc chi tiết

– **Input:** Hình ảnh $224 \times 224 \times 3$.

– **Convolution Layer 1:** Kernel 7×7 , stride 2, 64 filters.

– **Pooling Layer:** MaxPooling 3×3 , stride 2.

– **Residual Blocks:**

* Block 1: 3 Residual Blocks, $64 \rightarrow 256$ filters.

* Block 2: 4 Residual Blocks, $128 \rightarrow 512$ filters.

* Block 3: 6 Residual Blocks, $256 \rightarrow 1024$ filters.

* Block 4: 3 Residual Blocks, $512 \rightarrow 2048$ filters.

– **Fully Connected Layer:** 1000 lớp với Softmax.

4. Hàm mất mát

Cross-Entropy Loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

5. Lợi ích và cải tiến

– ResNet có thể đào tạo sâu hơn mà không bị mất gradient.

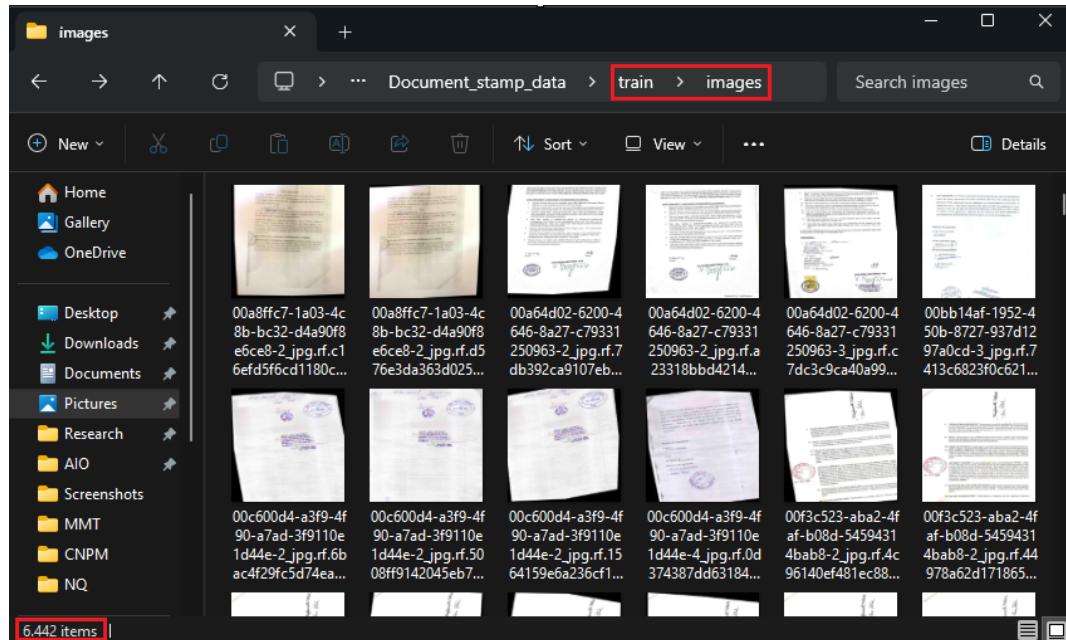
– Hiệu quả cao trong các bài toán thị giác máy tính.

– Áp dụng tốt cho cả phân loại hình ảnh và phát hiện đối tượng.

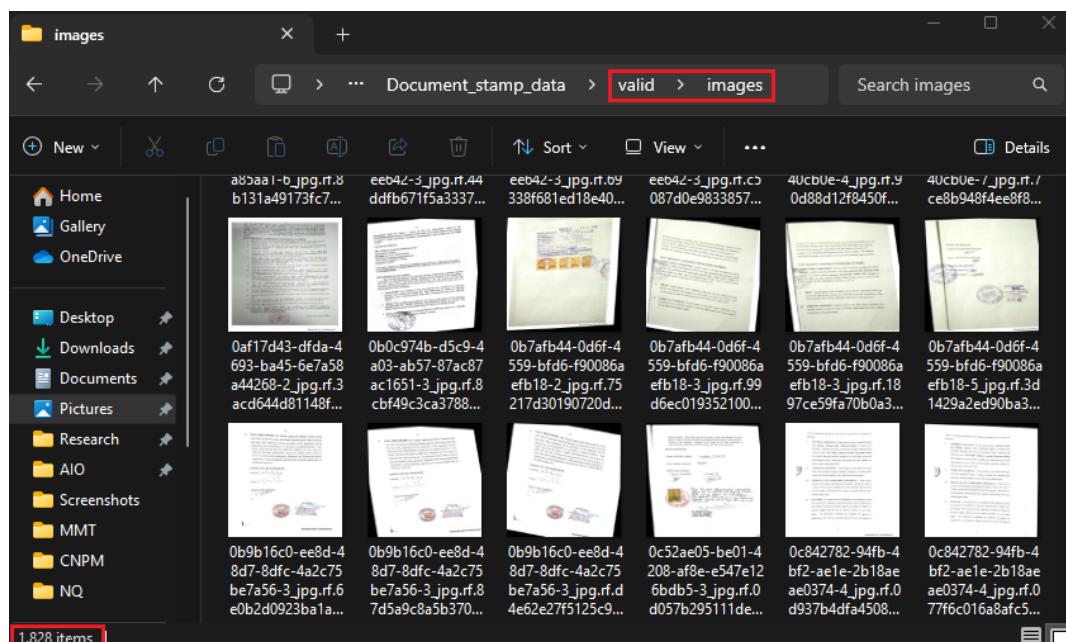
3 Dữ liệu được sử dụng trong đề tài

3.1 Dữ liệu stamp

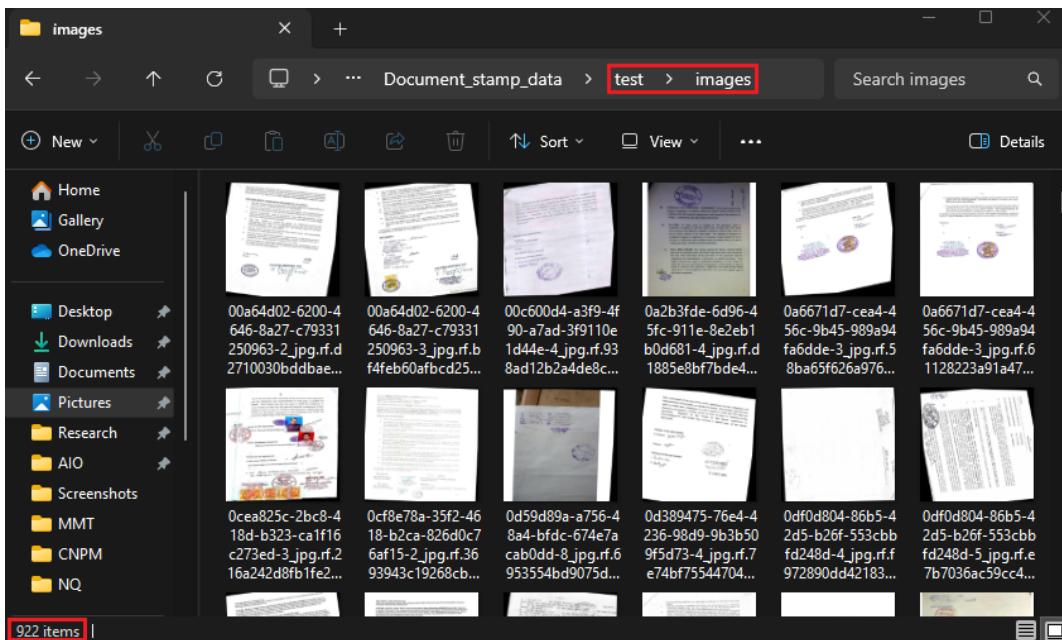
- Để huấn luyện mô hình YOLOv11 nhận diện được vị trí con stamp trên tài liệu nhóm sử dụng tổng cộng gần 10000 ảnh document đã được label để tiến hành huấn luyện.
- Trong đó được chia ra: 6442 ảnh dùng để train, 1828 cho tập valid và 922 ảnh cho tập test.



Hình 22: Stamp Train Data



Hình 23: Stamp Valid Data



Hình 24: Stamp Test Data

- Để huấn luyện kiểm tra so sánh xem con stamp có trong cơ sở dữ liệu của mình không, nhóm dùng một số con stamp mô phỏng. Mỗi một con stamp có 4-7 ảnh khác nhau ở nhiều góc hoặc cường độ sáng khác nhau của con stamp đó.



Hình 25: Stamp 1



Hình 26: Stamp 2

- Với đặc thù số lượng của mỗi con stamp không quá nhiều, nhóm hướng tới một mô hình cần phải giải quyết được vấn đề có thể học cách so sánh các class dữ liệu trong điều kiện khang hiếm. Nhận thấy khá tương đồng với các giải thuật nhận diện khuôn mặt, nhóm áp dụng chính mô hình đó vào bài toán này.

3.2 Dữ liệu chữ ký

4 Mô hình và Kết quả các mô hình được sử dụng cơ bản

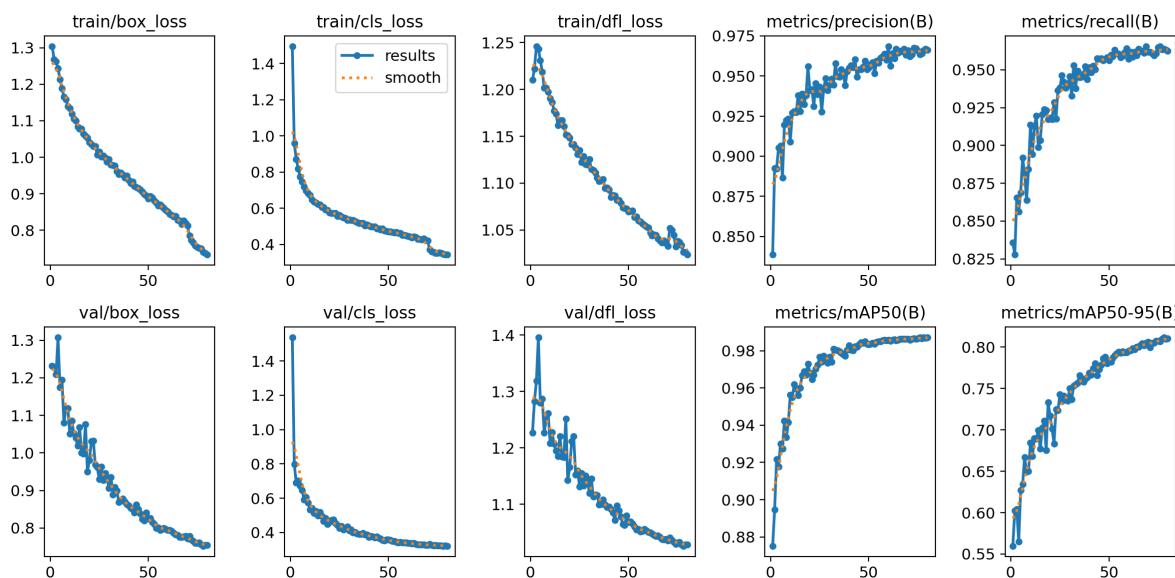
4.1 Nhận diện và phân loại stamp

Nhận diện stamp trên document

- Để nhận diện vị trí các con stamp nhóm sử dụng data đã giới thiệu ở phần 3.1 và model YOLOv11 đã giới thiệu ở phần 2.2.4 để giải quyết bài toán này.

```
1     from ultralytics import YOLO
2
3     model = YOLO("yolo11n.pt")
4
5     train_results = model.train(
6         data="Document_stamp_data\\data.yaml",
7         epochs=80,
8         imgsz=640,
9         device=1,
10    )
```

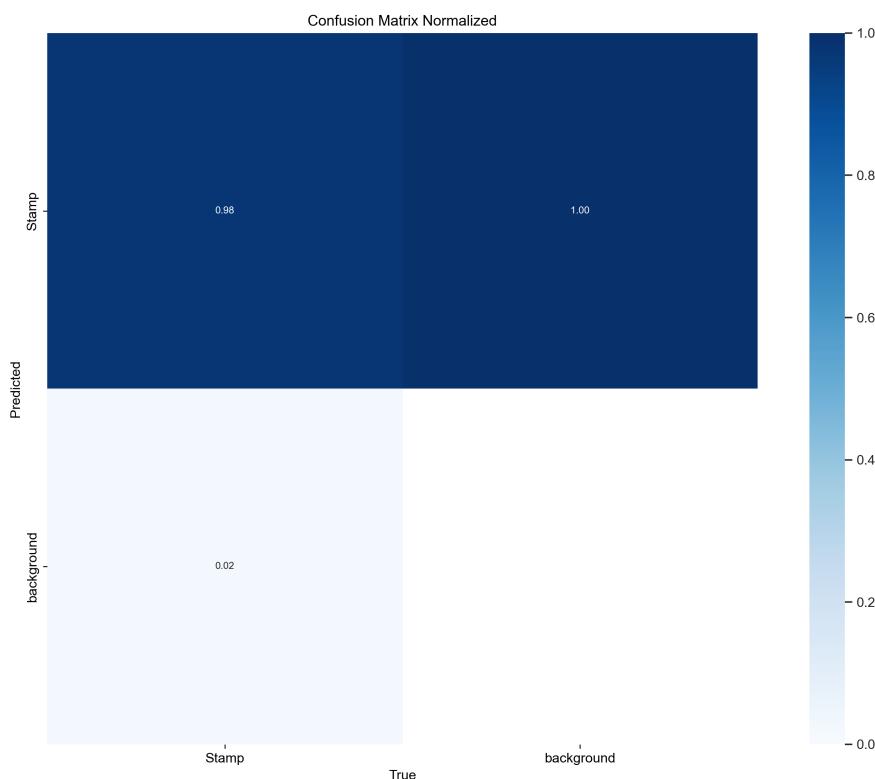
- Các kết quả mô hình sau khi train bằng YOLOv11:



Hình 27: Results Model YOLOv11

- Biểu đồ cung cấp thông tin chi tiết về quá trình huấn luyện và đánh giá mô hình, bao gồm các loại mất mát (loss) và các chỉ số hiệu suất.
- Nhìn chung mô hình có đường loss đi xuống ở cả 2 tập train và val nên có thể nói mô hình có khả năng học và hiệu quả học tập khá tốt.

- mAP@0.5-0.95 thấp hơn mAP@0.5: Điều này là bình thường nhưng cho thấy mô hình vẫn có thể cải thiện ở các ngưỡng IoU cao hơn



Hình 28: Confusion_Matrix_Normalized Model YOLOv11

- Ma trận Confusion đã được chuẩn hóa cho thấy:
 - Nhãn "Stamp" dự đoán đúng: 0.98 (98%).
 - Nhãn "Stamp" bị dự đoán sai thành "background": 0.02 (2%).
 - Nhãn "background" được dự đoán đúng: 1.00 (100%).
 - Nhãn "background" không có lỗi nhầm lẫn.
 - Một kết quả rất ấn tượng của mô hình.
- Các kết quả thực tế
 - Các kết quả thực nghiệm thực tế cho thấy hiệu suất vượt trội của mô hình trong việc nhận diện chính xác vị trí của các vật thể, thực hiện phân loại một cách rõ ràng và chính xác, đồng thời định vị các vật thể bằng cách tạo bounding box với độ chính xác cao.
 - Hơn nữa, mô hình còn thể hiện khả năng ấn tượng trong việc nhận diện đồng thời nhiều con dấu (stamp) xuất hiện trên cùng một tài liệu, ngay cả trong những trường hợp khó khăn như khi con dấu bị chồng chéo hoặc bị che khuất bởi chữ ký.

Bảng báo giá thiết kế: Hòa Casa - Biên Hòa

1. Quy trình làm việc và khối lượng công việc:

- a) **Giai đoạn 1:**
 - Sơ phác ý tưởng
 - Khảo sát đánh giá hiện trạng công trình
 - Phân tích giá yêu cầu khách hàng
 - Đề xuất ý tưởng sơ bộ:
 - . Mô hình ý tưởng 1/100 hoặc hình ảnh 3D
 - Thiết kế cơ sở
 - Mô hình tổng thể.
 - Mô hình bố trí các tầng. - Mô hình
 - Mô hình
- b) **Giai đoạn 2:**
 - Thiết kế nội thất
 - Mô hình bố trí thiết bị nội thất
 - Phối cảnh các không gian nội thất
 - Thiết kế cảnh quan
 - Mô hình bố trí
 - Phối cảnh các không gian cảnh quan
- c) **Giai đoạn 3:**
 - Thiết kế kĩ thuật thi công
 - Hồ sơ thiết kế kiến trúc - Hồ sơ thiết kế nội thất
 - Hồ sơ thiết kế kết cấu - Hồ sơ thiết kế cảnh quan
 - Hồ sơ thiết kế điện nước
- d) **Giai đoạn 4: Giám sát tác giả**
 - .Giải thích và làm rõ các tài liệu thiết kế công trình cho CDT, các nhà thầu khác để quản lý và thi công theo đúng thiết kế.
 - .Sửa đổi thiết kế đối với những nội dung chưa phù hợp với tiêu chuẩn, điều kiện thực tế của công trình.
 - .Trong quá trình thực hiện, khi có vướng mắc về thiết kế, đơn vị thiết kế phải có mặt tại hiện trường để giải quyết kịp thời. Phối hợp với CDT khi được yêu cầu để giải quyết các vướng mắc, phát sinh về thiết kế trong quá trình thi công xây dựng, xử lý những bất hợp lý trong thiết kế theo yêu cầu của CBT.
 - .Trong quá trình thực hiện, nếu thiết kế không phù hợp do lỗi nhà thiết kế, đơn vị thiết kế phải chịu trách nhiệm với thiết kế ban đầu.

2. Tiền độ công việc:

a) Giai đoạn 1: Sơ phác ý tưởng	20	Ngày làm việc
b) Giai đoạn 2: Thiết kế cơ sở	7	Ngày làm việc
c) Giai đoạn 3: Hồ sơ thiết kế KTTC	28	Ngày làm việc
d) Giai đoạn 4: Giám sát tác giả	-	Theo thực tế
Tổng tiền độ (Chưa kể giám sát)	90	Ngày làm việc

3. Tiền độ thanh toán :

Kí hợp đồng	100%	
a) Giai đoạn 1: Sơ phác ý tưởng	30%	Phi thiết kế
b) Giai đoạn 2: Thiết kế cơ sở	20%	Phi thiết kế
c) Giai đoạn 3: Hồ sơ thiết kế KTTC	20%	Phi thiết kế
d) Giai đoạn 4: Giám sát tác giả	10%	Phi thiết kế

4. Đơn giá thiết kế

a) Phản kiến trúc	40,000,000	vnd/gói
b) Phản nội thất	40,000,000	vnd/gói
c) Phản kết cấu	3,000,000	vnd/gói
d) Phản điện nước MEP	3,000,000	vnd/gói
e) Phản cảnh quan (<50m2)	15,000,000	vnd/gói
f) Phản giám sát tác giả	10,000,000	vnd/gói

Tổng phí thiết kế	111,000,000	vnd
Công tác phí (15 buổi)	7,500,000	
Tổng phí	118,500,000	vnd
Giảm giá	10,000,000	vnd
Tổng phí sau giảm	110,000,000	vnd

Stamp 0.87
CTA
KTS. Bùi Thế Long

(Một trăm mươi triệu đồng)

CTA

Hình 29: Detection Result 1



Hình 30: Detection Result 2



Xác định con dấu có trong cơ sở dữ liệu không



4.2 Nhận diện và phân loại chữ ký

5 Tổng kết