

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYỄN NGÔ TÍN - 1612700**

**NGUYỄN HỮU TÚ - 1612766**

**KHÓA LUẬN TỐT NGHIỆP**

**NHẬN DIỆN NỘI DUNG TRÊN THẺ CĂN CƯỚC  
CÔNG DÂN VIỆT NAM**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN**

**CHƯƠNG TRÌNH CHÍNH QUY**

**GIÁO VIÊN HƯỚNG DẪN**

**ThS. Lê Ngọc Thành**

**Ts. Lê Mai Tùng**

**Tp. Hồ Chí Minh, ngày 20 tháng 7 năm 2020**

# Lời cảm ơn

Nhóm xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Lê Ngọc Thành cùng TS. Lê Mai Tùng - công ty EyeQTech đã tận tình chỉ bảo, định hướng nhóm trong quá trình thực hiện luận văn này. Hai thầy đã tận tình cung cấp nhiều kiến thức, tài liệu cũng như các ý kiến góp ý, tạo điều kiện để nhóm thực hiện đề tài đạt kết quả tốt nhất.

Qua thời gian học tại trường Đại học Khoa học tự nhiên, nhóm xin chân thành gửi lời cảm ơn tới các thầy cô đã cung cấp những kiến thức quý giá, tạo điều kiện hỗ trợ trong suốt quá trình học tập để chúng em có những định hướng cho tương lai.

Do thời gian có hạn và khả năng của bản thân còn nhiều hạn chế. Nhóm đã nỗ lực trau dồi kiến thức và cố gắng thực hiện hết khả năng để hoàn thành đề tài một cách tốt nhất. Tuy nhiên nhóm khó có thể tránh khỏi những sai sót trong quá trình thực hiện đề tài này. Kính mong sự góp ý của quý thầy cô

Cuối cùng, nhóm xin gửi lời chúc sức khỏe và lời cảm ơn chân thành nhất đến các thầy cô

TP. HCM, ngày 20/7/2020

# Mục lục

LỜI CẢM ƠN	2
CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI	6
1.1. GIỚI THIỆU TÌNH HÌNH	6
1.2. CÁC GIẢI PHÁP ĐÃ CÓ	8
1.2.1. Tesseract và các cải tiến	8
1.2.2. BeeOCR	9
1.2.3. FPT AI Vision	10
1.3. LÝ DO LỰA CHỌN ĐỀ TÀI	11
1.4. MỤC TIÊU CỦA ĐỀ TÀI	11
1.5. PHẠM VI VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI	11
1.5.1. Phạm vi đề tài	11
1.5.2. Hướng phát triển của đề tài	12
CHƯƠNG 2: LÝ THUYẾT NỀN TẢNG	12
2.1. LÝ THUYẾT VỀ XỬ LÝ ẢNH	12
2.2. MẠNG NƠON NHÂN TẠO	16
2.2.1. Giới thiệu mạng nơron nhân tạo	16
2.2.2. Hàm kích hoạt (activation function)	18
2.2.3. Cách huấn luyện mạng nơron	22
2.3. MẠNG NƠON TÍCH CHẬP (CONVOLUTIONAL NEURAL NETWORK)	24
CHƯƠNG 3 GIẢI PHÁP	33
3.1. TỔNG QUAN GIẢI PHÁP	33
3.1.1. Cắt ảnh	33

3.1.2.	Nhận dạng vị trí của các thông tin trên thẻ căn cước	34
3.1.3.	Đọc thông tin từ mô hình nhận dạng ký tự quang học	36
3.1.4.	Sửa các lỗi chính tả ở bước 3.	37
3.2.	GIỚI THIỆU TÁC VỤ PHÁT HIỆN ĐỐI TƯỢNG TRÊN ẢNH, VIDEO SỐ(OBJECT DETECTION)	40
3.2.1.	R-CNN	40
3.2.2.	Fast R-CNN	41
3.2.3.	Faster R-CNN	43
3.2.4.	YOLO, YOLOv2, YOLOv3	45
3.2.5.	Single Shot MultiBox Detector(SSD)	47
3.3.	BỘ TRÍCH XUẤT TRONG CÁC MÔ HÌNH PHÁT HIỆN ĐỐI TƯỢNG	52
3.3.1.	MobileNet	52
3.3.2.	ResnetFpn	58
3.4.	MÔ HÌNH NHẬN DẠNG KÝ TỰ QUANG HỌC	64
3.5.	GIẢI PHÁP NHÓM CHỌN CHO TÁC VỤ PHÁT HIỆN ĐỐI TƯỢNG	64
<b>CHƯƠNG 4</b>	<b>CÀI ĐẶT GIẢI PHÁP</b>	<b>65</b>
4.1.	GIỚI THIỆU PYTHON VÀ TENSORFLOW	65
4.1.1.	Python	65
4.1.2.	Thư viện Tensorflow:	65
4.2.	CÀI ĐẶT HAI TÁC VỤ CẮT ẢNH VÀ NHẬN DIỆN VỊ TRÍ CÁC THÔNG TIN TRÊN THẺ CĂN CƯỚC	67
4.2.1.	Chuẩn bị dữ liệu	68
4.2.1.1.	Chuẩn bị ảnh	68
4.2.1.2.	Gán nhãn cho ảnh	69
4.2.1.3.	Tạo tệp dữ liệu huấn luyện cho mô hình	70
4.2.2.	Huấn luyện mô hình	71
4.3.	CÀI ĐẶT THƯ VIỆN PYTESSERACT VÀ NGÔN NGỮ VIỆT CHO PYTESSERACT ĐỂ ĐỌC THÔNG TIN	73

4.4.	CÀI ĐẶT SỬA CHÍNH TẢ CHO TÊN, ĐỊA CHỈ.	74
4.4.1.	Sửa chính tả cho tên	74
4.4.2.	Sửa chính tả cho địa chỉ	75
4.5.	TẠO MỘT API NHẬN VÀO MỘT ẢNH VÀ TRẢ RA CÁC THÔNG TIN TRÊN THẺ CĂN CƯỚC	76
<b>CHƯƠNG 5:</b>	<b>TỔNG KẾT</b>	<b>77</b>
5.1	KIẾN THỨC THU ĐƯỢC	77
5.2	SẢN PHẨM THU ĐƯỢC	77
5.3	ĐÁNH GIÁ ĐỘ CHÍNH XÁC CỦA SẢN PHẨM	77
5.4	PHƯƠNG HƯỚNG PHÁT TRIỂN VÀ NGHIÊN CỨU TRONG TƯƠNG LAI	80
<b>PHỤ LỤC</b>		<b>82</b>
	GIỚI THIỆU COLAB	82
<b>TÀI LIỆU THAM KHẢO</b>		<b>83</b>

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

## 1.1. Giới thiệu tình hình

Ngày nay, giấy tờ định danh cá nhân (chứng minh nhân dân, thẻ căn cước công dân) ngày càng được sử dụng rộng rãi trong nhiều lĩnh vực đời sống. Việc các tổ chức, cá nhân có nhu cầu thu thập, lưu trữ thông tin cá nhân của khách hàng, đặc biệt là thông tin trên các giấy tờ tùy thân ngày càng nhiều. Trong các thủ tục hành chính, chứng minh nhân dân/ căn cước công dân là loại giấy tờ không thể thiếu. Để phục vụ nhu cầu lưu trữ, thông tin thường được thu thập theo hai cách truyền thống:

- Khách hàng, người làm thủ tục sẽ điền các thông tin vào một mẫu đơn được chuẩn bị trước. Mẫu đơn này có thể là đơn bằng giấy hoặc form nhập liệu trên các website/ứng dụng máy tính. Khi số lượng người làm thủ tục nhiều, tại một thời điểm sẽ chỉ đủ không gian cho một số người điền. Sau khi điền xong, cần thêm nhân lực để kiểm tra những thông tin người đăng ký điền/nhập có khớp với thông tin trên giấy tờ hay không. Sai sẽ phải làm lại, điều này dẫn đến thời gian làm thủ tục bị kéo dài.
- Cách làm thứ hai giúp người làm thủ tục giảm thời gian chờ đợi. Thay vì phải điền vào các mẫu đơn hay biểu mẫu, người đăng ký sẽ nộp một bản photo giấy tờ tùy thân hoặc hình chụp rõ nét. Sau đó, giấy tờ photo sẽ được chuyển đến bộ phận nhập liệu và nhập vào máy tính. Cách làm này giúp người làm thủ tục giảm thời

gian chờ đợi, nhưng cần thêm nhân lực cho công việc nhập liệu. Việc nhập liệu một số lượng lớn thông tin trong thời gian dài dễ dẫn đến sai sót, nhập sai hoặc thiếu thông tin ghi trên giấy tờ.

Tại một số thời điểm, nhu cầu lưu trữ thông tin cá nhân của khách hàng, người làm thủ tục hành chính tăng cao trong thời gian ngắn. Ví dụ vào thời điểm sinh viên nhập học, nhu cầu làm thẻ ngân hàng để đóng học phí tăng cao. Các ngân hàng phải xử lý hàng ngàn hồ sơ trong một ngày. Ngân hàng nông nghiệp và phát triển nông thôn Việt Nam (Agribank) đã liên kết với hơn 40 trường đại học. Sau mỗi kì thi đại học, hàng chục ngàn sinh viên có nhu cầu làm thẻ ngân hàng. Nhân viên ngân hàng thu thập hàng chục ngàn bộ hồ sơ, bao gồm chứng minh nhân dân/ căn cước photo. Tiến hành nhập liệu, phân loại, xử lý và kiểm tra trong nhiều tháng. Tốn nhân lực và thời gian lao động rất lớn. Chính vì vậy để giảm thiểu chi phí, tăng độ chính xác và hiệu quả công việc. Nhu cầu về các ứng dụng nhập liệu tự động, tự động trích xuất thông tin từ giấy tờ tùy thân ngày một lớn.

Vì vậy các hệ thống, ứng dụng tự động trích xuất thông tin trên giấy tờ được ra đời để giải quyết vấn đề trên. Với sự phát triển của máy tính, nhu cầu giải quyết các bài toán nhận diện ký tự quang học (OCR) trên vật thể ngày càng nhiều. Thẻ căn cước công dân là loại thẻ ra đời sau, nhằm thay thế cho thẻ chứng minh nhân dân cũ. Hiện tại đã có nhiều nghiên cứu, ứng dụng ra đời nhằm tự động trích xuất thông tin trên hình ảnh thẻ căn cước công dân. Mỗi ứng dụng, phương pháp có những ưu điểm, hạn chế khác nhau. Việc phát triển một ứng dụng nhận diện ký tự quang học (OCR) với các loại giấy tờ định danh gặp những khó khăn không nhỏ. Những giấy tờ này chứa các thông tin cá nhân nên khó thu thập dữ liệu, dữ liệu không nhiều nên khó cải thiện được độ chính xác. Với việc nhu cầu trích xuất thông tin tự động ngày càng nhiều, dữ liệu đầu vào đa dạng (hình ảnh nhiều góc chụp, độ phân giải khác nhau, giấy tờ photo,...) Nhu cầu tăng tốc độ nhận diện, cải thiện độ chính xác của các ứng dụng là cần thiết

Trong khuôn khổ luận văn này, nhóm sẽ trình bày các kỹ thuật nhận diện nội dung trên thẻ căn cước công dân Việt Nam.

## **1.2. Các giải pháp đã có**

Để đáp ứng nhu cầu, cùng với sự phát triển của các mô hình nhận diện ký tự quang học (Optical character recognition - OCR), nhiều giải pháp đã được đề xuất, một số ứng dụng đã được xây dựng để giải quyết bài toán trên. Dưới đây là một số giải pháp tiêu biểu

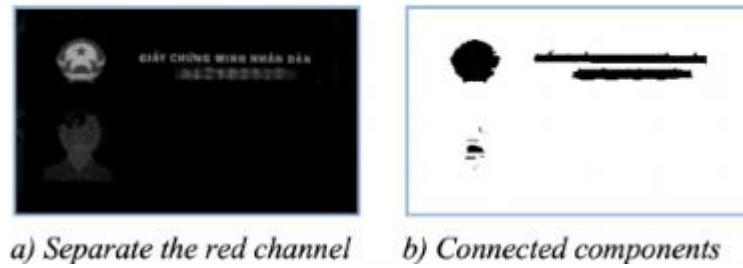
### **1.2.1. Tesseract và các cải tiến**

Tesseract là ứng dụng nhận diện ký tự đa ngôn ngữ, trích xuất nội dung văn bản từ các bản scan, hình chụp. Ứng dụng này cho kết quả chính xác cao khi nhận diện hình có chữ màu đen trên nền trắng. Bằng các thuật toán xử lý ảnh, ảnh chứng minh nhân dân/ căn cước công dân được biến đổi thành hình đen trắng với hai kênh màu. Sau đó đưa vào mô hình Tesseract để nhận diện

Mô hình này có thể cho kết quả chính xác với những hình được chụp đúng góc, chất lượng hình ảnh tốt. Nhưng không thể nhận diện với các hình chụp nghiêng, ngược. Đồng thời mô hình này chỉ trích xuất được toàn nội dung, toàn bộ chữ trong hình. Không trích xuất được theo mẫu

Để khắc phục vấn đề chỉ trích xuất toàn nội dung mà không trích xuất được theo mẫu. Khoa Công nghệ thông tin của Đại học Điện lực Hà Nội đã có một nghiên cứu về nhận diện các khu vực chứa nội dung theo form trên giấy chứng minh bằng các kỹ thuật xử lý ảnh [1]



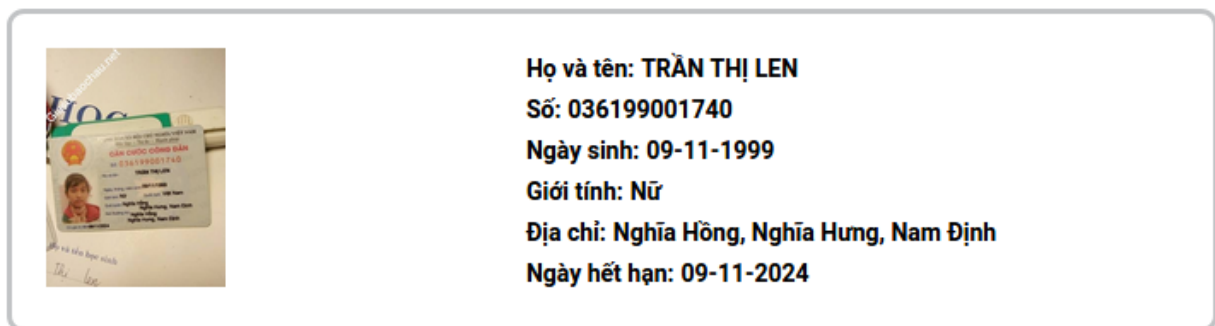


*Ảnh 1: Xác định vị trí các trường thông tin bằng kỹ thuật nhị phân hóa hình ảnh*

Phương pháp này sử dụng kỹ thuật xử lý ảnh để biến giấy chứng minh nhân dân/ căn cước công dân thành ảnh đen trắng, với phần chữ màu đen, nền màu trắng. Từ đó xác định được vị trí có chữ để cắt ảnh. Phương pháp này có thể trích xuất ra nội dung theo mẫu, nhưng với hình ảnh bị mờ, khi qua phần tiền xử lý (preprocessing) có thể mất nội dung. Vấn đề ảnh chụp nghiêng, đảo ngược vẫn chưa thể xử lý được


### 1.2.2. BeeOCR

BeeOCR là hệ thống hỗ trợ nhận diện thẻ chứng minh nhân dân với độ chính xác khá tốt. Tuy nhiên, với thẻ căn cước công dân, ứng dụng này còn chưa hoàn thiện. Ứng dụng chưa hỗ trợ để đọc quê quán, quốc tịch.



*Ảnh 2: Một thẻ căn cước được trích xuất từ BeeOCR*

Với những ảnh bị mờ, độ phân giải thấp, một số ký tự bị đọc sai, chưa có cơ chế sửa lỗi chính tả. Khiến nội dung đọc được chỉ gần đúng



**Họ và tên:** TRẦN QUỐC NANH

**Số:** 031200009676

**Ngày sinh:** 26-06-2030

**Giới tính:** Nam

**Địa chỉ:** Thôn A Hưng Bình Pthuận Nguyên Hà Phòng

**Ngày hết hạn:** 20-06-2028

*Ảnh 3: Một ảnh BeeOCR nhận diện sai*

### 1.2.3. FPT AI Vision

FPT AI Vision là một ứng dụng được FPT cho ra đời 2 năm gần đây, nhận diện được tất cả nội dung mặt trước của thẻ căn cước với độ chính xác lên tới 94,6 %



<b>Full name</b>	ĐINH THỊ THU PHƯƠNG
<b>Number</b>	036199006541
<b>Birthday</b>	18/04/1999
<b>Gender</b>	NỮ
<b>Nationality</b>	VIỆT NAMVIỆT NAM
<b>Hometown</b>	LIÊM HẢI, TRỰC NINH, NAM ĐỊNH
<b>Address</b>	LIÊM HẢI, TRỰC NINH, NAM ĐỊNH
<b>Date of Expiry</b>	18/04/2024

*Ảnh 4: Nội dung được FPT AI trích xuất từ thẻ căn cước công dân*

Tuy nhiên, ứng dụng này không thể cài đặt sử dụng trên máy của khách hàng mà thông qua API. Vấn đề của giải pháp này là chi phí khá cao với những ứng dụng sử dụng dưới 10000 yêu cầu(request) mỗi tháng thì giá là 1000đ/1 yêu cầu. Mỗi yêu cầu chỉ bao gồm 1 ảnh chưa 1 thẻ căn cước trên đó.

### 1.3. Lý do lựa chọn đề tài

Nhằm áp dụng các kiến thức tích lũy trong quá trình học tập vào một sản phẩm có ý nghĩa thực tế, giải quyết được nhu cầu trong đời sống hiện tại. Nhóm lựa chọn đề tài “Nhận diện thông tin trên thẻ căn cước công dân Việt Nam” nhằm cải tiến độ chính xác cũng như khắc phục một số hạn chế của những ứng dụng đã có trên thị trường. Ứng dụng này có thể áp dụng vào nhiều lĩnh vực trong đời sống nhằm mục đích lưu trữ, tăng tốc độ, hiệu quả làm việc so với cách làm truyền thống.

### 1.4. Mục tiêu của đề tài

- Trình bày các kiến thức nền tảng trong bài toán nhận diện nội dung trên thẻ căn cước công dân.
- Cải thiện được độ chính xác của các giải pháp đã có cho bài toán trên.
- Tạo ra được một API nhận vào một ảnh chứa thẻ căn cước trả về thông tin trên thẻ căn cước với độ chính xác ở mức chấp nhận được với mức phí rẻ hơn(có thể là miễn phí).
- Tạo một ứng dụng demo cách sử dụng API.

### 1.5. Phạm vi và hướng phát triển của đề tài

#### 1.5.1. Phạm vi đề tài

Xây dựng một ứng dụng đọc thông tin trên thẻ căn cước công dân Việt Nam, chỉ đọc thông tin văn bản trên mặt trước của thẻ, không nhận diện ảnh của chủ thẻ.

Dữ liệu hình ảnh căn cước công dân có nhiều định dạng, kích thước, độ phân giải, độ sáng khác nhau gây ra nhiều khó khăn cho việc nhận dạng. Do đó cần giới hạn một số điều kiện:

- Hình ảnh có độ phân giải cao, rõ ràng và đọc được bằng mắt thường
- Góc nghiêng không quá 90 độ so với phương ngang

- Thẻ căn cước không bị mờ, trầy xước, nhòe, chữ còn rõ ràng
- Không bị nhiễu sáng, chói do ánh sáng khi chụp
- Thẻ căn cước có chứa các nội dung như mã số định danh, tên, ngày sinh, giới tính, quốc tịch, quê quán, nơi đăng ký thường trú.

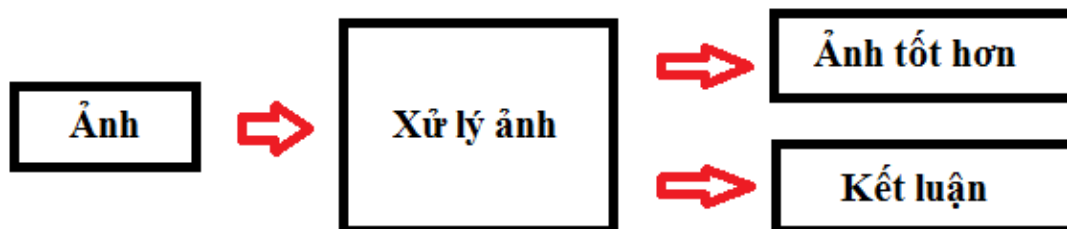
### 1.5.2. Hướng phát triển của đề tài

Xây dựng một ứng dụng áp dụng vào thực tế, kết hợp với các mô hình nhận diện các giấy tờ định danh khác như chứng minh nhân dân cũ, bằng lái xe, hộ chiếu,... thành một hệ thống nhận dạng thông tin trên giấy tờ định danh hoàn chỉnh, có thể đọc thông tin ở cả mặt trước lẫn sau. Có thể xây dựng thêm một mô hình nhận diện khuôn mặt để xác định giấy tờ có chính chủ hay không.

## Chương 2: Lý thuyết nền tảng

### 2.1. Lý thuyết về xử lý ảnh

Thị giác máy tính đóng vai trò rất quan trọng trong tương tác người - máy. Việc xử lý ảnh hiện nay được xử lý nhanh chóng, cho kết quả tốt nhờ sự phát triển của phần cứng máy tính cũng như các card đồ họa. Mục đích của xử lý ảnh là từ ảnh đầu vào, qua quá trình thao tác sẽ cho ra một ảnh tốt hơn ở một số khía cạnh, phục vụ cho các tác vụ sau đó. Hoặc từ quá trình xử lý cho ra một kết luận về ảnh



*Ảnh 5: Mô hình vai trò của xử lý ảnh*

➤ Một số khái niệm:

- Số hóa ảnh:

Số hóa ảnh là sự biến đổi một ảnh trong thực tế thành một tập hợp điểm tương đương với ảnh thật về vị trí, màu sắc, không gian. Mỗi điểm trong tập điểm được gọi là một Pixel, trong không gian hai chiều, mỗi pixel tương ứng với một cặp tọa độ  $(x,y)$

- Điểm ảnh (pixel): Là một phần tử của ảnh tại tọa độ  $(x,y)$  với màu nhất định
- Độ phân giải (Resolution): Là mật độ điểm ảnh trên một ảnh hiển thị
- Mức xám của ảnh:

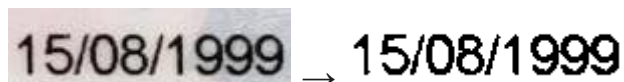
Một điểm ảnh có hai đặc trưng cơ bản: tọa độ  $(x,y)$  và độ xám của nó. Mức xám của điểm ảnh là cường độ sáng của một điểm

➤ Một số kỹ thuật xử lý ảnh được sử dụng

- Nhị phân hóa hình ảnh

Nhị phân hóa hình ảnh là kỹ thuật biến đổi ảnh có nhiều kênh màu sang ảnh nhị phân. Trong bài toán nhận diện chữ viết, nhị phân hóa hình ảnh rất quan trọng. Vai trò của phương pháp này là phân chia ảnh thành hai phần, phần nền trắng và phần chữ màu đen. Phương pháp này tính toán độ sáng thích hợp của ảnh, sau đó chọn một ngưỡng. Bước cuối cùng là chuyển tất cả các pixel có độ sáng lớn hơn ngưỡng thành một giá trị cố định (thường là trắng), các giá trị bé hơn thành một giá trị khác (thường là đen)

Ảnh dưới minh họa kỹ thuật nhị phân hóa ngày ảnh chứa ngày sinh được cắt ra từ thẻ căn cước:



- Lọc nhiễu:

Nhiều là tập hợp các điểm sáng thừa trên ảnh, bị gây ra bởi nhiều nguyên nhân như chói sáng khi chụp, máy ảnh độ phân giải thấp. Loại bỏ nhiễu là một vấn đề quan trọng và thường gặp trong các bài toán nhận dạng, trích xuất đặc trưng từ ảnh. Nhiễu ảnh có nhiều loại: Nhiễu đốm, nhiễu vệt, nhiễu đứt nét:



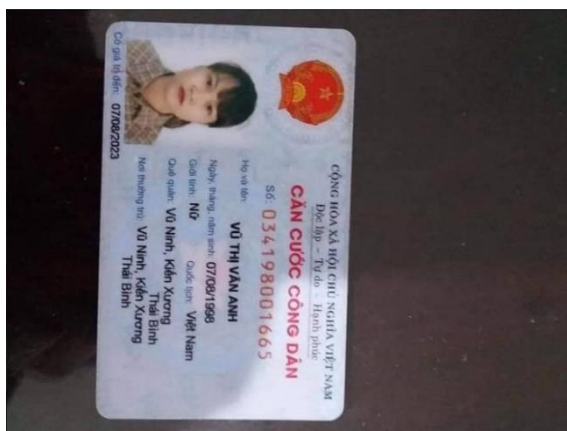
*Ảnh 6: Ảnh bị nhiễu sáng sau khi nhị phân hóa*

Để loại bỏ nhiễu ra khỏi hình ảnh, có nhiều phương pháp lọc. Tuy nhiên với các ảnh có độ nhiễu lớn, các phương pháp lọc thường dùng như lọc trung bình, lọc trung vị tỏ ra kém hiệu quả. Một phương pháp khác được sử dụng để loại nhiễu đó là bỏ các vùng không liên thông ra khỏi hình

- **Cắt ảnh**

Cắt ảnh là một trong những kỹ thuật cần thiết để chuẩn hóa hình ảnh. Với hình ảnh có nhiều phần dư thừa, việc xác định các đường biên và lấy phần trọng tâm là hết sức cần thiết. Tăng độ chính xác trong quá trình trích xuất và nhận diện, đồng thời loại bỏ các nhiễu không cần thiết, những vùng này có thể làm sai lệch kết quả.

Trong bài toán nhận diện thẻ căn cước công dân, kỹ thuật cắt ảnh được sử dụng để cắt ảnh khi đã xác định được bốn góc của thẻ căn cước



- Chỉnh mức xám:

Kỹ thuật này thực hiện nội suy các mức xám trung gian bằng kỹ thuật nội suy, mục đích để tăng độ mịn cho ảnh:



- Xoay ảnh:

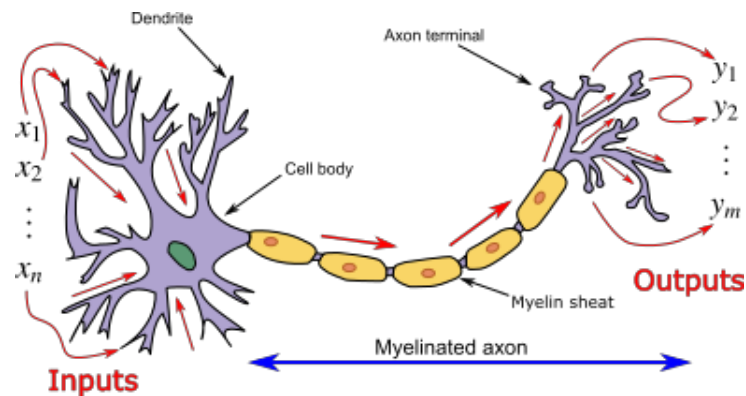
Do kỹ thuật chụp ảnh, hình ảnh của thẻ căn cước bị nghiêng một góc so với phương ngang. Điều này gây khó khăn cho việc trích xuất ra các đặc trưng, thuộc tính, đối tượng của hình ảnh. Đôi khi không thể nhận diện được hoặc nhận diện không chính xác. Một hướng xử lý là phải tính toán lại tọa độ của các đối tượng

Có nhiều kỹ thuật để xoay ảnh trở về trạng thái chuẩn, trong phạm vi luận văn này. Việc xoay ảnh được thực hiện dựa vào việc trích xuất các góc của thẻ căn cước. Từ tọa độ trích xuất được, qua một số phép biến đổi toán học, tính được góc nghiêng và xoay ảnh để được ảnh tốt hơn

## 2.2. Mạng nơron nhân tạo

### 2.2.1. Giới thiệu mạng nơron nhân tạo

Mạng nơron nhân tạo là mô hình được xây dựng dựa trên mạng nơron sinh học, mô phỏng cách tiếp nhận và xử lý thông tin của bộ não người.



Ảnh 7: Mô hình tế bào thần kinh người

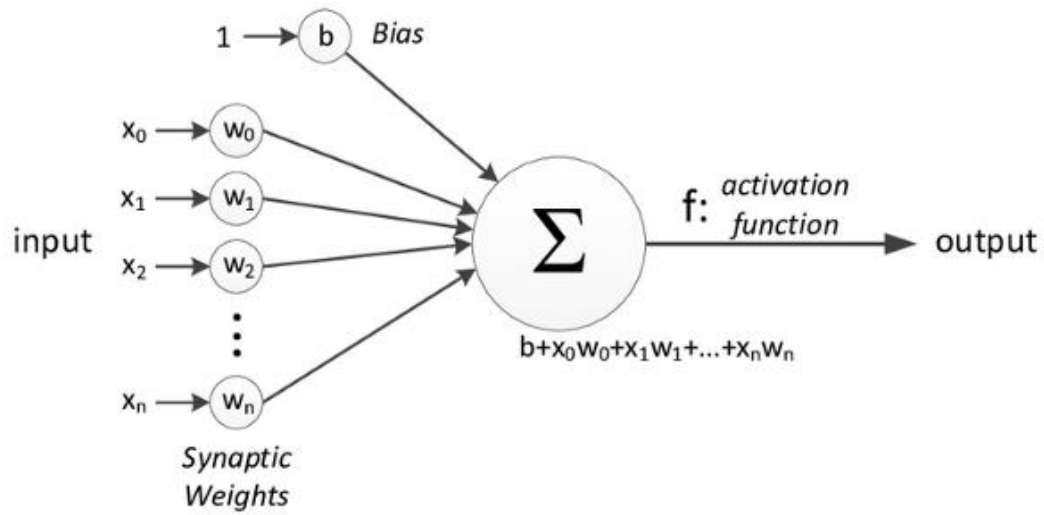
Mô hình hoạt động của tế bào thần kinh (neural) là các tín hiệu đầu vào (input) được tiếp nhận, được xử lý và cho ra các tín hiệu đầu ra (output), các tín hiệu đầu ra này lại là đầu vào của nơron tiếp theo. Cứ tương tự vậy, thông tin khi truyền qua nhiều tầng nơron sẽ cho ra đầu ra cuối cùng.

Từ mô hình xử lý của tế bào thần kinh, mô hình mạng nơron nhân tạo được xây dựng.

Mô hình sau là mạng nơron 1 lớp (single layer neural network) hay còn gọi là perceptron.

Là mạng nơron nhân tạo đơn giản nhất





Ảnh 8: Cấu trúc của một perceptron

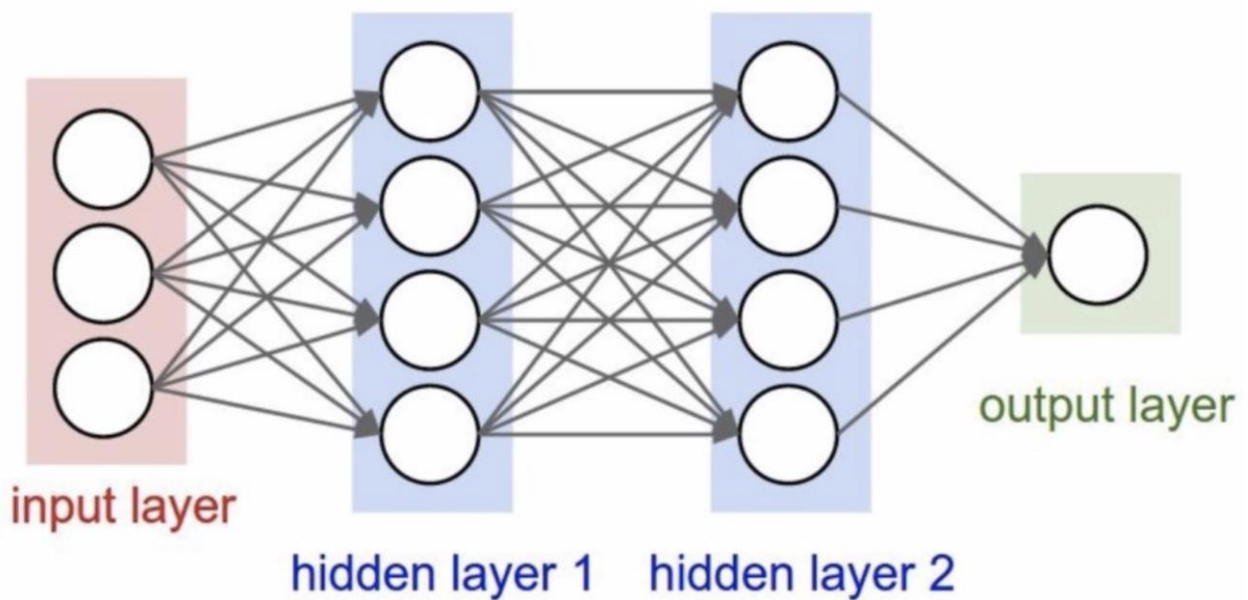
Mô hình đơn giản này bao gồm nhiều tín hiệu đầu vào (input)  $x_0, x_1, x_2, \dots, x_n$ . Các giá trị này độc lập với nhau. Mỗi đầu vào được nhân với một trọng số liên kết (connection weight)  $w_0, w_1, w_2, \dots, w_n$ . Sau đó cộng với một giá trị bias  $b$ . Kết quả được đưa vào một hàm kích hoạt  $f$  (activation function) và cho ra một đầu ra duy nhất. Trọng số thể hiện mức độ quan trọng của một tín hiệu đầu vào.

Công thức toán học  $x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \dots + x_n \cdot w_n = \sum x_i \cdot w_i$

Kết quả sau khi áp dụng hàm kích hoạt  $\phi(x_i \cdot w_i)$

Một mạng nơron bao gồm nhiều lớp (layer), tầng đầu tiên là tầng input, tầng cuối cùng là tầng output, các tầng ở giữa gọi là tầng ẩn (hidden layer).

Số lượng hidden layer không giới hạn. Đầu ra của tầng trước là đầu vào của tầng tiếp theo



Ảnh 9: Mô hình một mạng nơron đầy đủ

### 2.2.2. Hàm kích hoạt (activation function)

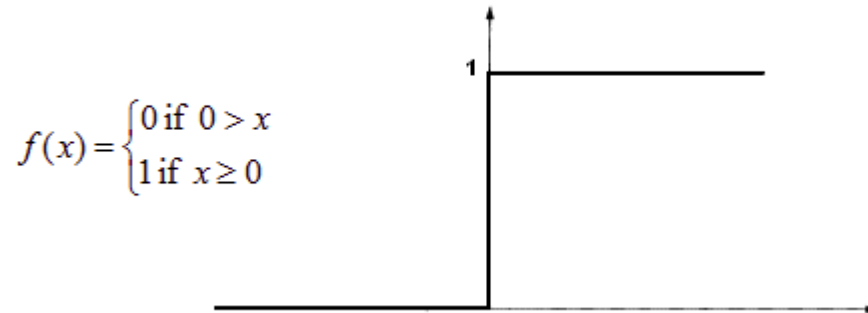
Hàm kích hoạt là một thành phần quan trọng trong mạng nơron nhân tạo. Mục đích chính của nó là biến đổi các tín hiệu đầu vào thành đầu ra, đầu ra này sẽ được sử dụng làm đầu vào cho lớp tiếp theo.

Nếu không sử dụng hàm kích hoạt, khi đó mạng nơron sẽ đơn thuần là một hàm tuyến tính. Và các bài toán mà mối quan hệ giữa đầu vào và đầu ra là một hàm phi tuyến tính không thể được giải quyết bằng mạng nơron nhân tạo.

Mục đích của một mạng nơron nhân tạo là tìm mối liên hệ giữa đầu vào và đầu ra, mối liên hệ này thường là các hàm phi tuyến tính. Mạng nơron sẽ được huấn luyện để tìm ra mối liên hệ này thông qua các trọng số liên kết. Để biến các biến đổi tuyến tính thành phi tuyến tính, hàm kích hoạt là một thành phần cực kỳ quan trọng. Dưới đây là những hàm kích hoạt thường sử dụng trong các mô hình nhận diện, xử lý ảnh

- Hàm kích hoạt Threshold

Hàm kích hoạt Threshold là hàm lựa chọn nhị phân

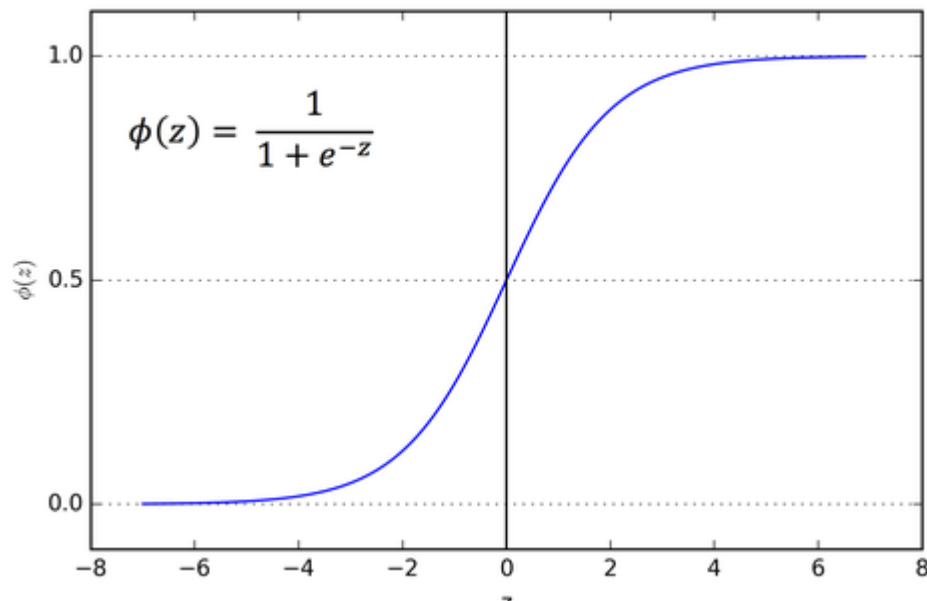


Ảnh 10: Đồ thị hàm Threshold

Nếu giá trị đầu vào lớn hoặc nhỏ hơn một ngưỡng (threshold), neuron sẽ được kích hoạt và truyền tín hiệu đến lớp tiếp theo. Hàm này thường được sử dụng trong các bài toán đầu ra có hai lựa chọn. Nhưng không thể sử dụng trong trường hợp đầu ra có nhiều giá trị

- Hàm sigmoid

Hàm Sigmoid là một hàm toán học có đồ thị là đường cong hình chữ S, vùng giá trị nằm trong khoảng từ 0 tới 1. Hàm Sigmoid thường dùng để dự đoán xác suất của một đầu ra

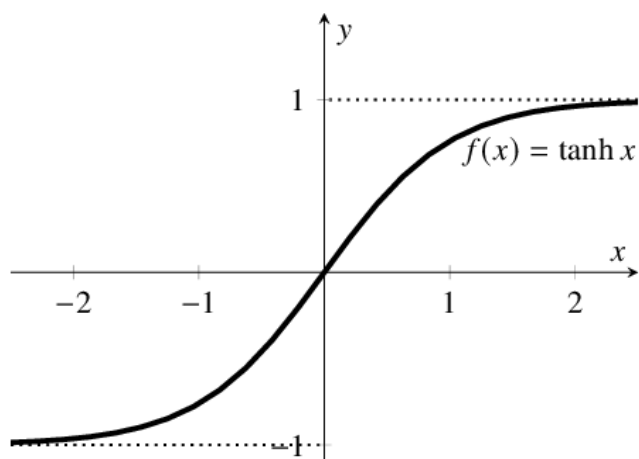


Ảnh 11: Đồ thị hàm Sigmoid

Hàm Sigmoid là một hàm khả vi, nghĩa là chúng ta có thể ước lượng được "độ dốc" của đường cong tại một điểm. Hạn chế của hàm sigmoid là giá trị đầu vào là giá trị âm rất nhỏ và dương rất lớn thì kết quả đầu ra không có sự khác biệt nhiều (xấp xỉ 0 hoặc 1). Gây cản trở cho quá trình huấn luyện

- Hàm tanh

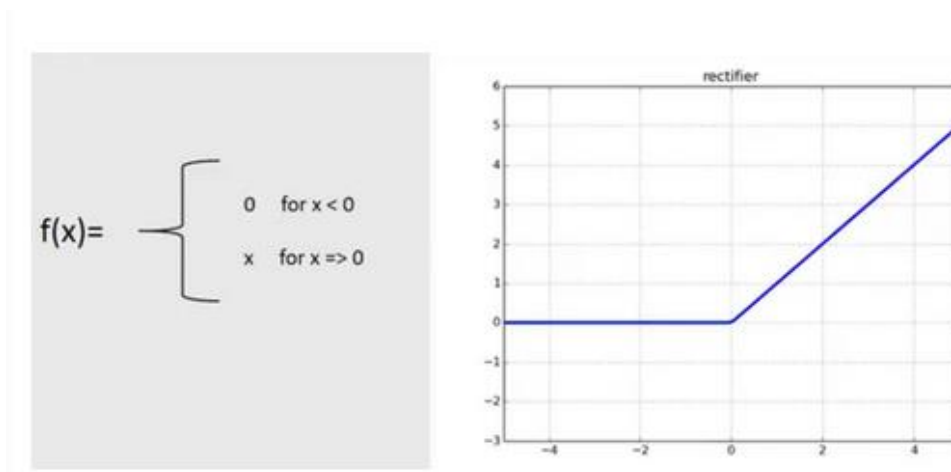
Hàm tanh là hàm có hình dạng tương tự như hàm Sigmoid, vùng giá trị nằm trong khoảng  $[-1, 1]$ , cho phép đầu ra có giá trị . Ưu điểm của hàm này là hàm phi tuyến tính, đầu vào giá trị âm cho đầu ra giá trị âm và ngược lại. Điều này ít gây cản trở hơn trong quá trình huấn luyện



Ảnh 12: Đồ thị hàm tanh

- Hàm Rectified Linear Units (ReLU)

Hàm ReLU là hàm được sử dụng nhiều trong các mạng nơ-ron nhân tạo (ANN) và mạng nơ-ron tích chập (CNN). Vùng giá trị nằm trong khoảng  $(0, +\infty)$



Ảnh 13: Đồ thị hàm ReLU

Nếu đầu vào là giá trị dương sẽ cho đầu ra là chính nó, ngược lại cho ra giá trị 0. Hạn chế của hàm này nếu đầu vào toàn giá trị dương, hàm trở thành hàm tuyến tính.

Hàm ReLu đã được chứng minh là làm quá trình huấn luyện nhanh hơn các hàm khác. Hàm ReLu có nhiều biến thể khác nhau như Noisy ReLu, Leaky ReLu, ELUS,....

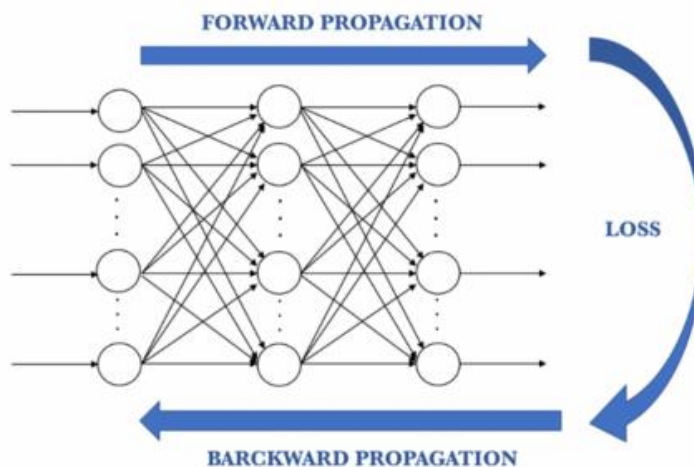
### 2.2.3. Cách huấn luyện mạng nơron

Cách học của mạng nơron tương tự như cách học trong thực tế. Thực hiện một hành động, điều chỉnh qua các lần thực hiện cho tới khi đạt được kết quả đúng. Để đánh giá sự khác nhau giữa kết quả thực tế và kết quả mạng nơron dự đoán. Khái niệm hàm mất mát (~~Cost function~~) được định nghĩa



Hàm mất mát: hàm số để tính toán sự sai khác giữa kết quả thực tế và giá trị mạng nơron tính toán được. Với mỗi tầng trong mạng nơron, hàm mất mát phân tích và điều chỉnh trọng số (weight) và ngưỡng giá trị (threshold). Mục tiêu là biến đổi để giảm giá trị của hàm mất mát. Giá trị của hàm mất mát càng nhỏ thì kết quả dự đoán của mạng nơron càng gần giá trị thực tế.

Khi có sự sai lệch giữa kết quả dự đoán và kết quả thực tế, trọng số sẽ được điều chỉnh. Và dữ liệu đầu vào được đưa vào mạng nơron có trọng số mới, một hàm mất mát mới được tạo ra. Quá trình được lặp đi lặp lại cho đến khi giá trị của hàm mất mát giảm xuống thấp nhất có thể



Ảnh 14: Mô phỏng quá trình huấn luyện mạng nơron

Quá trình này được gọi là lan truyền ngược(Back Propagation). Các trọng số được điều chỉnh sau mỗi lần dự đoán sao cho đến khi hàm mất mát đạt giá trị nhỏ nhất.

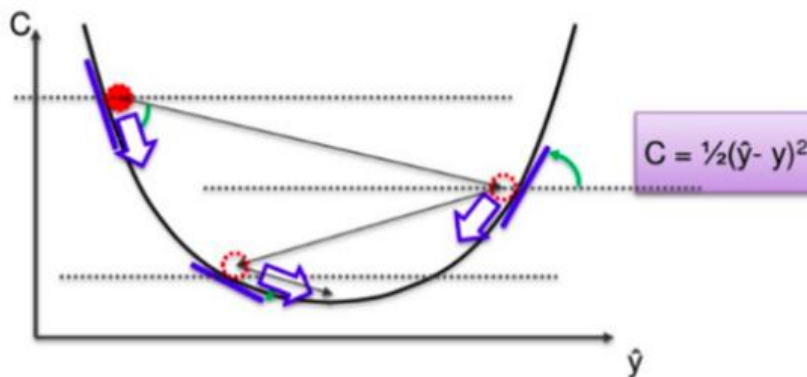
Phương pháp dùng để điều chỉnh trọng số đối với mạng noron nhiều lớp là phương pháp Gradient Descent

### ➤ Gradient descent

Đây là phương pháp lặp lại tối ưu theo từng bước (iterative improving) và tìm giá trị cực tiểu của hàm mất mát, từ đó điều chỉnh trọng số cho thích hợp. Gradient là độ dốc của hàm mất mát. Công thức của Gradient descent được mô phỏng dưới dạng:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} f(\theta)$$

trong đó  $\theta$  là tập các biến cần cập nhật,  $\eta$  là tốc độ học (*learning rate*),  $\nabla f(\theta)$  là Gradient của hàm mất mát  $f$  theo tập  $\theta$ .



Ảnh 15: Mô phỏng gradient descent

Trong phương pháp này, mục tiêu là tìm các giá trị để hàm đạt giá trị cực trị. Nếu hàm mất mát không phải là hàm lồi sẽ dẫn tới các giá trị cực đại/cực tiểu địa phương. Tuy nhiên trong phần lớn các bài toán thực tế, hàm mất mát là hàm lồi.

Việc lặp lại, điều chỉnh gradient descent dừng lại khi giá trị của hàm mất mát đủ nhỏ hoặc hàm mất mát không đổi sau một số bước liên tục.

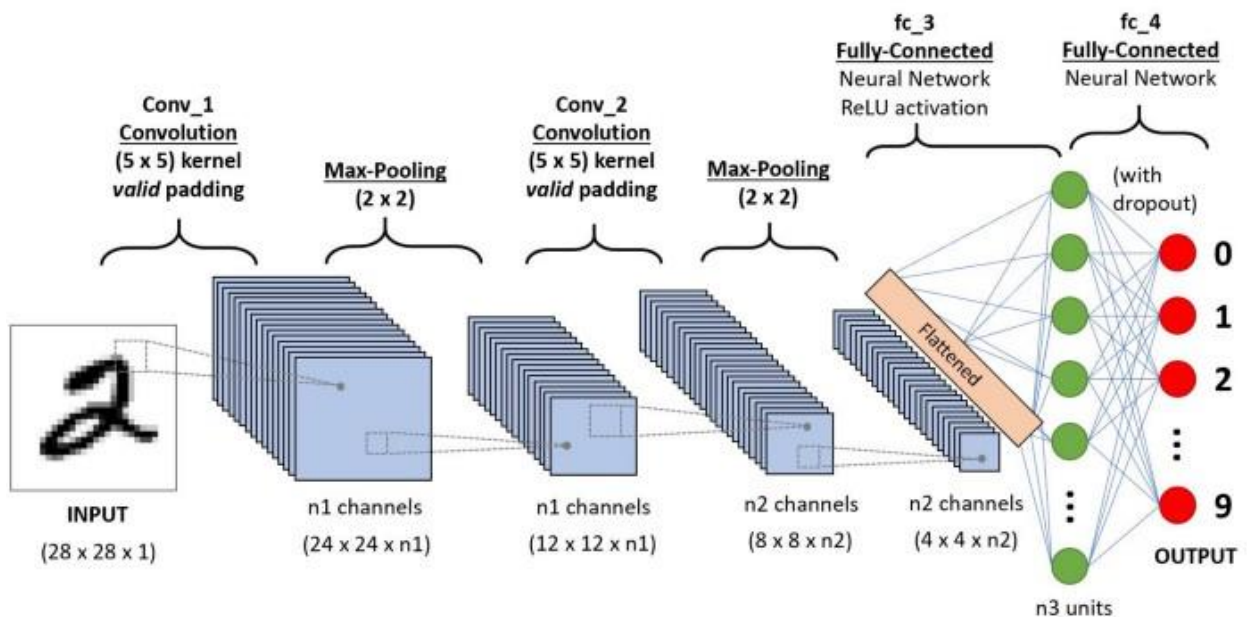
Trong luận văn này, các mô hình học máy (machine learning) chủ yếu là các mô hình liên quan đến xử lý hình ảnh và ngôn ngữ. Trong bài toán xử lý ảnh, dữ liệu đầu vào là hình ảnh có độ phân giải lớn, với ba kênh màu. Từ hình ảnh ban đầu, các giá trị pixel sẽ được đưa vào mạng nơron. Giả sử với hình ảnh có độ phân giải  $500 \times 500$  px và ba kênh màu RGB. Số lượng node của lớp input sẽ là  $500 \times 500 \times 3 = 750000$ . Điều này khiến số lượng phép tính trong quá trình huấn luyện sẽ rất lớn, làm giảm tốc độ và tài nguyên. Vì vậy mô hình Convolutional Neural Network (CNN) được áp dụng để tăng tốc độ và giảm khối lượng tính toán

### **2.3. Mạng nơron tích chập (Convolutional Neural Network)**

Trong các tác vụ liên quan đến thị giác máy tính (Computer Vision), mạng nơron thông thường có một vấn đề. Đó là việc tất cả các tầng đều là các tầng kết nối đầy đủ. Giả sử đối với một ảnh kích thước  $224 \times 224 \times 3$ . Vậy để đưa tất cả thông tin của ảnh vào mô hình thì lớp đầu vào của mô hình phải có  $224 \times 224 \times 3 = 150528$  nút (node). Sau đó ở tầng kết nối đầy đủ đầu tiên có 1000 nút thì số lượng tham số giữa hai tầng đầu tiên sẽ là 150528000. Vậy thì nếu kích thước ảnh tăng lên cộng với việc mô hình có nhiều lớp thì số lượng tham số sẽ tăng lên cực lớn, khiến cho việc tính toán trở nên rất chậm và vì vậy mạng nơron tích chập ra đời để giải quyết vấn đề này.

Mạng nơron tích chập là một thuật toán học sâu (Deep Learning) được sử dụng trong các bài toán xử lý ảnh. Cấu trúc của CNN là một mạng nơron có các lớp ẩn tiêu biểu như: Các lớp tích chập (convolutional layer), lớp tổng hợp (pooling layer), lớp kết nối đầy đủ (fully connected layer).

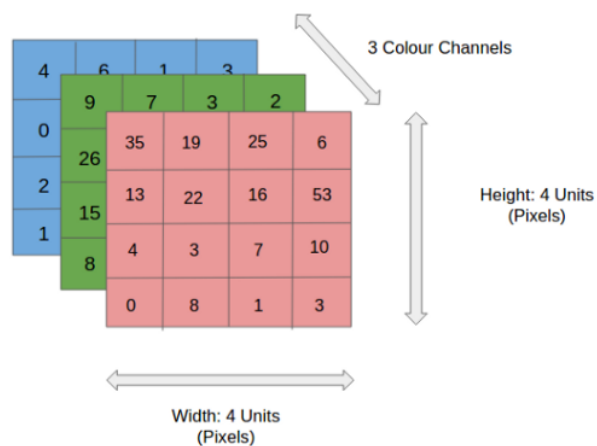




Ảnh 16: Mô hình của một mạng nơ-ron tích chập

Mạng nơ-ron tích chập nhận vào một bức ảnh, được đưa qua các lớp tích chập, tổng hợp, kết nối đầy đủ để trích xuất ra các đặc trưng nhằm các mục đích nhận diện, phân biệt hình ảnh.

### ➤ Ảnh đầu vào



Ảnh 17: Mô phỏng cấu trúc của một ảnh

Một bức hình đầu vào của mạng nơron tích chập thường là ảnh RGB với ba kênh màu cơ bản: đỏ, xanh lá, xanh dương. Được lưu trữ ở dạng một Tensor ba chiều. Với một bức ảnh có độ phân giải lớn (VD: 7680x4320x3), số lượng pixel đưa vào mạng nơron sẽ rất lớn, tốn nhiều tài nguyên lẫn thời gian tính toán. Lớp tích chập có vai trò giảm kích thước bức hình xuống nhưng không làm mất các thuộc tính quan trọng của bức hình, từ đó vẫn đảm bảo các dự đoán chính xác dù giảm khối lượng tính toán

### ➤ Lớp tích chập

Là lớp đầu tiên, dùng để trích xuất các đặc trưng của ảnh. Lớp tích chập đảm bảo duy trì mối quan hệ giữa các pixel trong ảnh bằng cách sử dụng các bộ lọc(filter).

Ảnh đầu vào là một Tensor ba chiều  $h \times w \times d$ . Trong đó  $h$ ,  $w$  là kích thước bức ảnh,  $d$  là số kênh màu

Một bộ lọc là một ma trận  $f_h \times f_w \times d$  có kích thước nhỏ hơn

Phép nhân giữa hai ma trận được thực hiện để cho ra kết quả đầu ra

Ví dụ: Hình ảnh đầu vào là một ma trận 5x5 với các pixel có hai giá trị 0 hoặc 1. Một ma trận 3x3 được sử dụng làm kernel:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

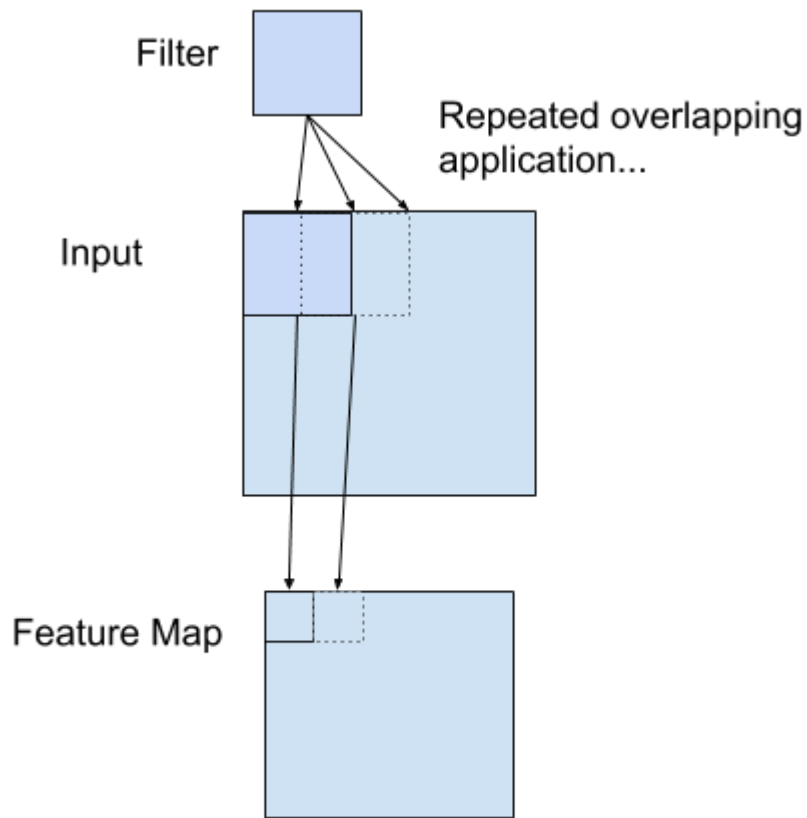
5 x 5 – Image Matrix



1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix








Phép nhân giữa hai ma trận cho ra một kết quả gọi là bản đồ đặc trưng, dùng để trích xuất ra các đặc trưng của ảnh:



*Ảnh 18: Mô phỏng quá trình trích xuất đặc trưng bằng bộ lọc*

Bộ lọc lần lượt trượt trên ảnh đầu vào theo từng bước, tại mỗi bước. Phép nhân giữa hai ma trận cùng kích thước được thực hiện để cho ra một giá trị duy nhất, các bước được thực hiện lặp lại cho đến khi bộ lọc trượt qua hết các phần tử của ma trận. Kết quả đầu ra là một bản đồ đặc trưng

Lớp tích chập sử dụng nhiều bộ lọc có thể trích xuất ra những đặc trưng khác nhau của ảnh như góc, đường thẳng, đường cong. Hình ảnh dưới minh họa các kết quả khác nhau khi áp dụng nhiều bộ lọc:

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Ảnh 19: Kết quả khi áp dụng nhiều bộ lọc cho ảnh

### ➤ Khắc phục nhược điểm của mạng nơron thông thường

Mạng nơron tích chập khắc phục nhược điểm của mạng nơron thông thường bằng hai ý tưởng chính là vùng tiếp nhận cục bộ(local receptive field) và chia sẻ trọng số(shared weight and bias).

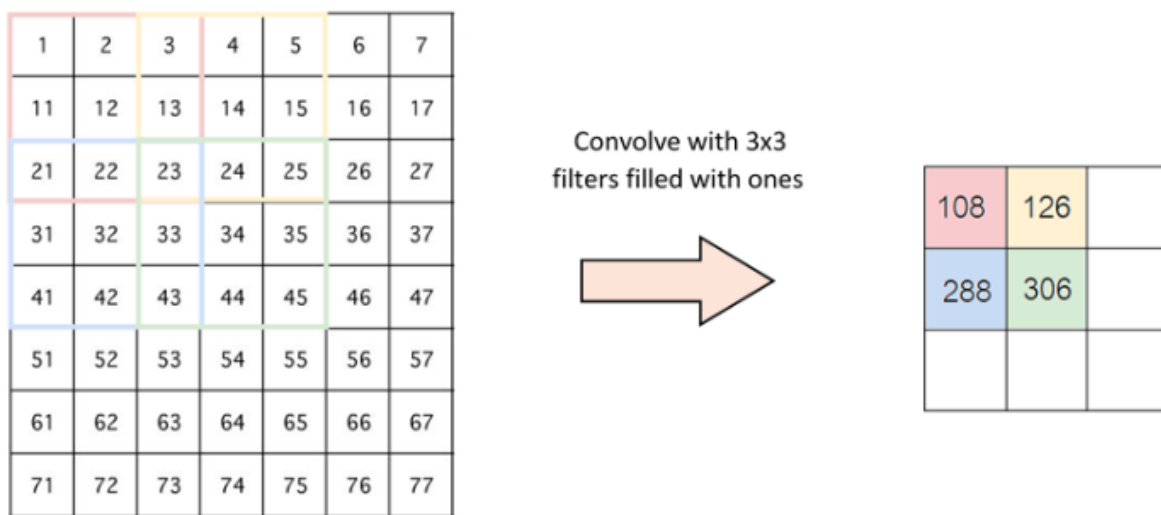
Giả sử đối với ảnh đầu vào có kích thước  $28 \times 28$ . Với mạng nơron thông thường thì tất cả  $28 \times 28 = 784$  nút này sẽ đưa vào lớp kết nối đầy đủ cùng một lúc, giả sử số nút trong tầng đầu tiên là 50 thì số tham số là  $784 \times 50 = 39200$  tham số và đầu ra của tầng này là 50 giá trị mới ứng với 50 nút. Với mạng nơron tích chập lớp tích chập đầu tiên có một bộ lọc

$5 \times 5$ . Với cơ chế trượt của bộ lọc trên ảnh để tính ra bản đồ đặc trưng thì mỗi vùng  $5 \times 5$  pixel (giống như mỗi vùng  $3 \times 3$  ở hình trên) được nhân với bộ lọc được gọi là vùng tiếp nhận cục bộ. Mạng nơron tích chập đưa từng vùng này vào để nhân với bộ lọc chứ không đem tất cả vào như mạng nơron thông thường. Kết quả là tất cả các vùng tiếp cận cục bộ chỉ nhân với một bộ lọc duy nhất và với bộ lọc  $5 \times 5$  cùng 1 tham số bias thì lớp tích chập này chỉ cần 26 tham số. Bản đồ đặc trưng có được khi đi qua lớp tích chập này với bước nhảy là 1, padding là 0(padding và bước nhảy sẽ được giải thích sau) có kích thước là  $24 \times 24$ .

Rõ ràng là lớp tích chập đã giảm được số lượng tham số rất nhiều so với lớp kết nối đầy đủ( từ 39200 xuống còn 26)

### ➤ Bước nhảy (Stride)

Bước nhảy (stride) là số pixel mà bộ lọc “trượt” trên ma trận đầu vào sau mỗi phép tính.



Ảnh 20: Ảnh minh họa cách Convolutional hoạt động với bước nhảy là 2

### ➤ Padding

Trong một số trường hợp, bộ lọc sẽ không tương thích với ma trận đầu vào. Vùng chưa được so sánh của ảnh đầu vào kích thước nhỏ hơn bộ lọc, khi đó có hai cách để xử lý:

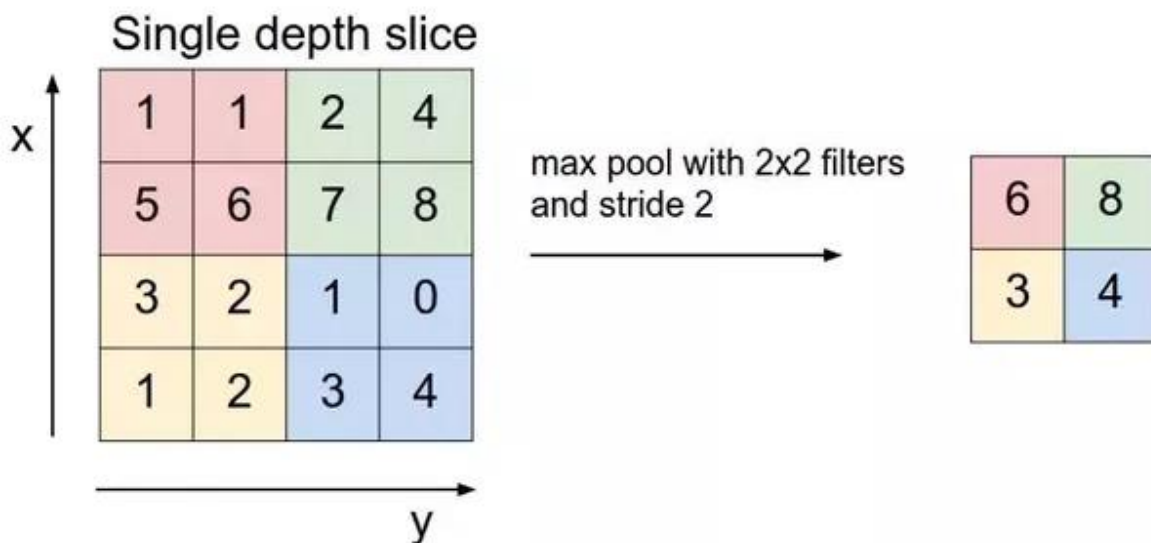
- Thêm các pixel có giá trị 0 (zero padding) để bộ lọc và ảnh tương thích
- Xóa bỏ những phần không tương thích ra khỏi ảnh

### ➤ Lớp tổng hợp (Pooling Layer)

Khi ảnh đầu vào có số lượng pixel lớn, số lượng tham số nhiều. Việc tính toán trở nên phức tạp. Tầng Pooling có vai trò giảm số lượng tham số xuống. Giảm chiều lẫn kích thước hình ảnh nhưng vẫn giữ được những thông tin, đặc tính quan trọng của hình để đảm bảo các tác vụ sau cho ra kết quả chính xác. Phương pháp tổng hợp có nhiều cách khác nhau:

- Max Pooling
- Average Pooling
- Sum Pooling

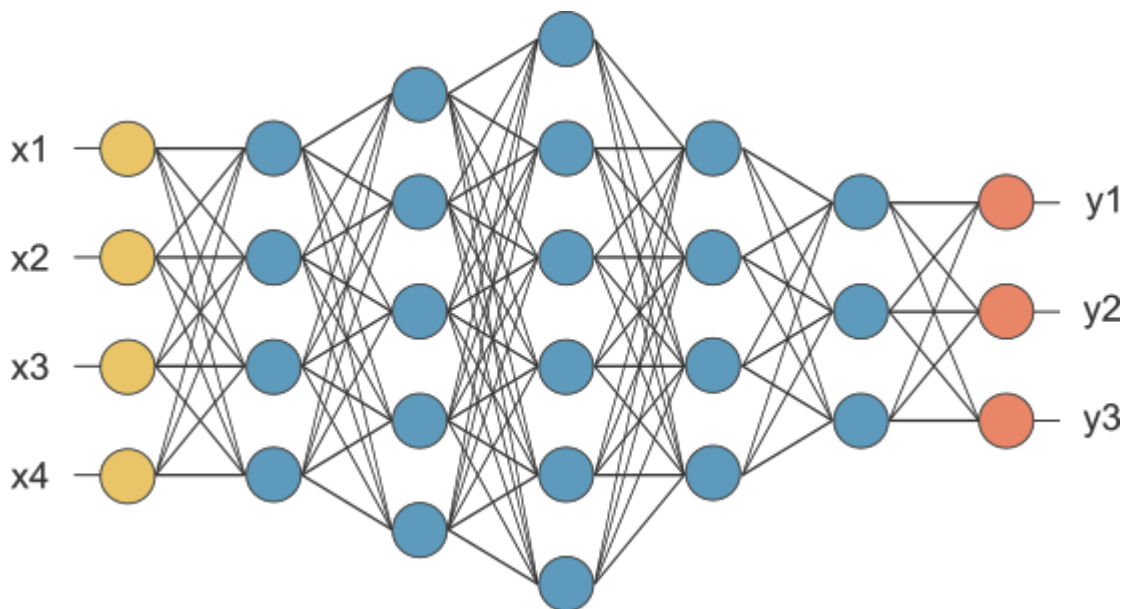
Trong đó Max pooling là phương pháp thường được sử dụng nhiều nhất. Max pooling lấy ra giá trị lớn nhất. Ngoài ra còn có thể lấy giá trị trung bình của các phần tử (Average Pooling) hoặc tổng giá trị (Sum pooling).



Ảnh 21: Ảnh minh họa phương pháp Max Pooling với bộ lọc . Bước nhảy 2

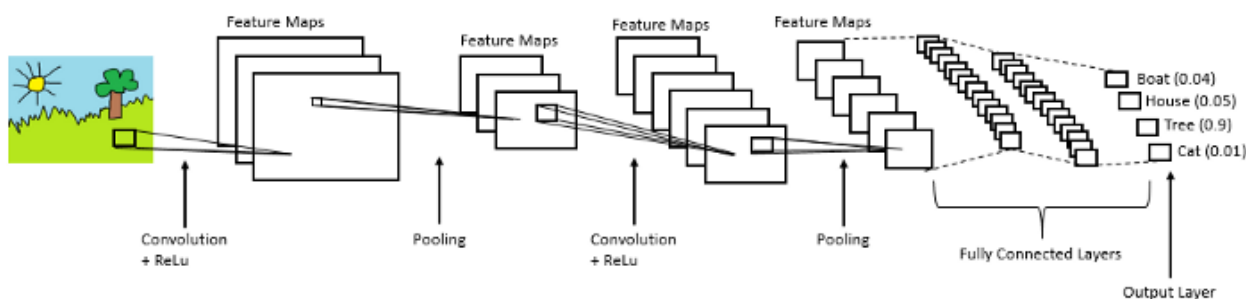
### ➤ Tầng kết nối đầy đủ (Fully Connected Layer)

Ở tầng này, các ma trận đầu ra của các tầng trước sẽ được làm phẳng thành vector một chiều và đưa vào tầng kết nối (Connected layer). Tầng kết nối là một mạng nơ-ron



Ảnh 22: Tầng kết nối đầy đủ

Ở hình minh họa trên: bản đồ đặc trưng được làm phẳng và chuyển thành vector  $(x_1, x_2, x_3, \dots)$ . Các đặc trưng của hình được trích xuất từ các tầng trước kết hợp với nhau thành mô hình huấn luyện. Qua nhiều tầng ẩn, ở tầng cuối cùng. Một hàm kích hoạt được sử dụng để cho ra kết quả cuối cùng tùy theo bài toán



Ảnh 23: Kiến trúc đầy đủ của một mạng nơ-ron tích chập:

Một số mô hình kiến trúc CNN đã được xây dựng để phục vụ giải quyết những bài toán khác nhau. Một số mô hình tiêu biểu:

- LeNet
- AlexNet
- VGGNet
- GoogleNet
- ResNet
- ZFNet

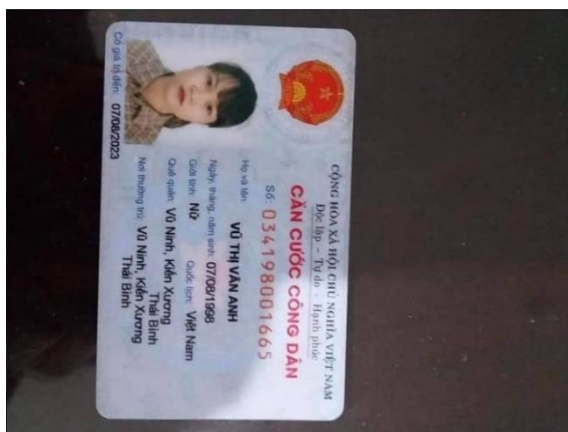


# Chương 3 Giải pháp

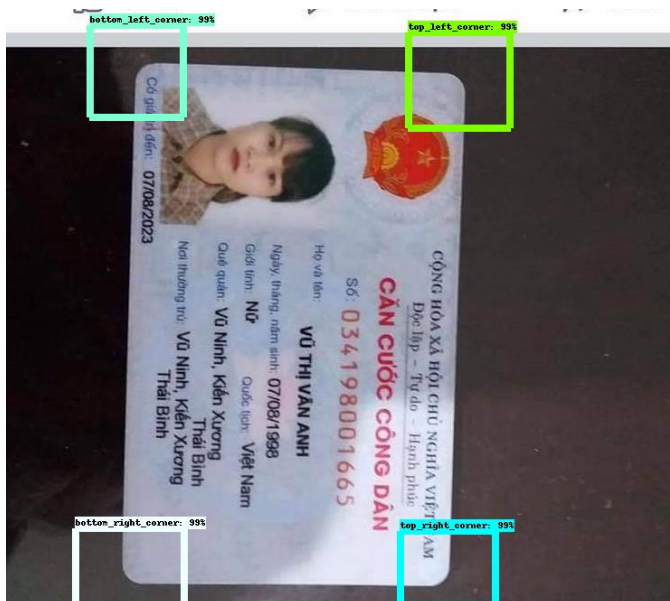
## 3.1. Tổng quan giải pháp

### 3.1.1. Cắt ảnh

Từ ảnh đầu vào dùng một mô hình nhận dạng(object detection model ) để nhận dạng(sẽ có phần giải thích về object detection và các model ở dưới) 4 góc của thẻ căn cước , từ vị trí 4 góc đó cắt thẻ ra khỏi nền(background)



Ảnh 24: Ảnh đầu vào



Ảnh 25: Ảnh được nhận diện 4 góc



Ảnh 26: Ảnh sau khi cắt khỏi nền

### 3.1.2. Nhận dạng vị trí của các thông tin trên thẻ căn cước

Các thông tin trên thẻ căn cước bao gồm:

- number: số thẻ căn cước
- name: tên chủ thẻ
- dob: ngày tháng năm sinh
- genre: giới tính
- national: quốc tịch
- hometown: quê quán
- address: nơi thường trú



Ảnh 27: Ảnh đầu vào



Ảnh 28: Ảnh kết quả nhận dạng

Như hình trên thì ta có thể thấy được là ở các trường tên, quê quán, nơi thường trú chúng tôi không nhận diện nguyên một dãy mà tách riêng lẻ từng từ một. Có hai lý do cho vấn đề trên:

- Thứ nhất là có những trường hợp không tìm được một hộp giới hạn(bounding box) phù hợp để chứa các đối tượng này



Ảnh 29: Trường hợp không thể tìm được khung giới hạn

Nếu hộp giới hạn chứa các ký tự khác ngoài các ký tự cần thiết thì khiến cho bước đọc thông tin sẽ bị sai lệch quá nhiều, khó cho quá trình sửa lỗi chính tả sẽ được trình bày ở những bước sau

- Thứ hai là việc tách ra sẽ mang lại lợi ích ở những trường hợp ảnh mờ, việc nhận dạng khó khăn sẽ bị mất một vài từ trong địa chỉ chứ không mất hoàn toàn nếu chỉ có một hộp giới hạn bao quanh địa chỉ

Sau khi có được vị trí của các đối tượng trên, đối với những đối tượng chỉ được tách ra thành nhiều đối tượng như tên, quê quán, nơi thường trú thì ta cần sắp xếp lại thứ tự. Tiếp theo ta sẽ cắt các ảnh chứa các đối tượng này và đưa vào mô hình nhận dạng ký tự quang học (Optical Character Recognition - OCR) để đọc thông tin

### 3.1.3. Đọc thông tin từ mô hình nhận dạng ký tự quang học

Đã có nhiều mô hình hỗ trợ nhận dạng ký tự từ hình ảnh cho độ chính xác khá cao. Tuy nhiên, đối với ngôn ngữ có dấu như tiếng Việt, việc nhận dạng cho ra kết quả có độ chính xác vẫn đang là một vấn đề gây nhiều khó khăn với các mô hình phổ biến hiện tại. Trong

các mô hình nhận diện tiếng Việt phổ biến, Tesseract là mô hình được sử dụng rộng rãi. Tesseract vốn là mô hình được dùng để nhận diện nội dung từ hình ảnh scan của văn bản. Tesseract cho kết quả chính xác khá cao với hình ảnh chữ đen trên nền trắng. Qua một vài kỹ thuật xử lý ảnh, hình ảnh trên thẻ căn cước vốn có nền trắng có thể biến đổi thành hình ảnh tương tự như hình ảnh scan văn bản. Từ đó áp dụng mô hình nhận diện của tesseract mang lại độ chính xác cao.

Ở bước hai, mỗi đối tượng trích xuất được là hình chữ nhật chứa một từ. Việc cần làm trước khi đưa hình ảnh chứa một từ qua mô hình Tesseract là xác định thứ tự các từ, để sau quá trình nhận diện ký tự sắp xếp lại thành nội dung chính xác



*Ảnh 30: Ba đối tượng cùng loại được nhận diện*

Với ba đối tượng nhận diện được cùng thuộc lớp "name". Dựa vào tọa độ của các đỉnh hình chữ nhật, thứ tự đúng của các hình chữ nhật được xác định và lưu lại. Sau bước nhận diện ký tự, sắp xếp lại nội dung đúng là "Bùi Minh Đức" thay vì thứ tự sai như "Đức Bùi Minh"

#### **3.1.4. Sửa các lỗi chính tả ở bước 3.**

Việc đọc sai chính tả ở mô hình nhận dạng ký tự quang học là không thể tránh khỏi, đặc biệt là đối với tiếng Việt vì dấu rất dễ bị sai bởi những ảnh bị nhiễu(noise), mờ, ... Vì vậy cần có bước chỉnh sửa lại lỗi chính tả để thông tin đọc được chính xác hơn.

##### **➤ Sửa ở tên**

Giả sử từ sai chính tả cần phải sửa là W và từ được chọn để sửa cho W là C. Đối với mỗi W ta không thể biết chắc chắn được chính xác được C là từ nào( ví dụ như từ “Tron” thì

có thể sửa thành “Trọng”, “Trang”, “Trần” ...). Vì vậy hướng tiếp cận của nhóm sẽ là hướng tính xác suất và chọn ra giá trị lớn nhất.

Xác suất cần tính ở đây là  $P(C|W)$ . Theo công thức Bayes thì:

$$P(C|W) = \frac{P(W|C) * P(C)}{P(W)}$$

Vì  $P(W)$  với mỗi  $W$  là như nhau nên ta xem như là  $P(C|W) = P(W|C) * P(C)$ . Lý do việc tách từ việc tính một xác suất  $P(C|W)$  ra thành 2 thành phần  $P(W|C)$  và  $P(C)$  là vì thật ra  $P(C|W)$  đã chứa hai yếu tố. Lấy ví dụ một từ “Nguyen”, bây giờ có 2 từ được cho là có thể sửa được là “Nguyễn” và “Nguyên”, vậy thì bây giờ  $P(C|W)$  nào cao hơn. “Nguyễn” thì có vẻ ít sai lệch hơn nhưng nếu đây là họ thì xác suất ra họ “Nguyễn” là cao hơn, nếu là tên thì xác suất ra tên “Nguyên” là cao hơn, rõ ràng cần phải quan tâm đến xác suất của từ được chọn để sửa ( $P(C)$ ). Vậy việc tách ra thành 2 phần  $P(W|C)$  và  $P(C)$  là hợp lý vì tách thành hai thành phần riêng biệt sẽ giúp ta dễ xử lý hơn.  $P(W|C)$  sẽ tính được bằng cách tính độ khớp của  $W$  so với  $C$ ,  $P(C)$  sẽ tính được bằng số lần  $C$  xuất hiện trong từ điển các từ đúng chính tả.

Như đã đề cập ở trên thì rõ ràng nếu là họ thì  $P(\text{“Nguyễn”})$  sẽ cao hơn  $P(\text{“Nguyên”})$ . Tuy nhiên nếu là tên thì  $P(\text{“Nguyên”})$  sẽ cao hơn  $P(\text{“Nguyễn”})$ . Vì vậy nhóm chia thành ba từ điển khác nhau, một dành cho họ, một dành cho tên và còn lại là tên lót. Dữ liệu tên thì nhóm thu thập bằng cách lấy danh sách trúng tuyển ở các trường đại học trên toàn quốc, bộ dữ liệu nhóm thu thập được bao gồm khoảng 50000 tên.

### ➤ Sửa ở quê quán, nơi thường trú

Dữ liệu ở dạng file JSON như hình bên dưới

```

"92": {
  "name": "Cần Thơ",
  "slug": "can-tho",
  "type": "thanh-pho",
  "name_with_type": "Thành phố Cần Thơ",
  "code": "92",
  "quan-huyen": {
    "916": {
      "name": "Ninh Kiều",
      "type": "quan",
      "slug": "ninh-kieu",
      "name_with_type": "Quận Ninh Kiều",
      "path": "Ninh Kiều, Cần Thơ",
      "path_with_type": "Quận Ninh Kiều, Thành phố Cần Thơ",
      "code": "916",
      "parent_code": "92",
      "xa-phuong": {
        "31117": {
          "name": "Cái Khế",
          "type": "phuong",
          "slug": "cai-khe",
          "name_with_type": "Phường Cái Khế",
          "path": "Cái Khế, Ninh Kiều, Cần Thơ",
          "path_with_type": "Phường Cái Khế, Quận Ninh Kiều, Thành phố Cần Thơ",
          "code": "31117",
          "parent_code": "916"
        },
        "31120": {
          "name": "An Hòa",
          "type": "phuong",
          "slug": "an-hoa",
          "name_with_type": "Phường An Hòa",
          "path": "An Hòa, Ninh Kiều, Cần Thơ",
          "path_with_type": "Phường An Hòa, Quận Ninh Kiều, Thành phố Cần Thơ",
          "code": "31120",
          "parent_code": "916"
        }
      }
    }
  }
}

```

*Ảnh 31: Cấu trúc lưu trữ địa chỉ*

Giải pháp mà nhóm đưa ra để sửa lỗi chính tả ở chuỗi thông tin địa chỉ là nhóm sẽ phân tách chuỗi thành các chuỗi con chứa thông tin từng cấp của địa chỉ như tỉnh, thành phố rồi đến quận huyện tiếp theo là xã phường. Sau đó sẽ đi tìm độ sai lệch nhỏ nhất của các chuỗi con này với các thông tin dữ liệu hành chính Việt Nam ở trên. Từ đó tìm ra một địa chỉ đúng. Đầu tiên sẽ lấy chuỗi thông tin tỉnh, thành phố ra tính độ lệch với 64 tỉnh thành phố trên cả nước sau đó chọn ra được tỉnh, thành phố phù hợp nhất. Sau khi có được tỉnh, thành phố thì ta sẽ có được danh sách quận huyện của thành phố đó, tiếp tục lấy chuỗi



quận huyện đi so sánh với danh sách này và tìm ra được quận huyện phù hợp. Tiếp tục như vậy ta sẽ tìm được đến cấp xã phường.

## 3.2. Giới thiệu tác vụ phát hiện đối tượng trên ảnh, video số(Object Detection)

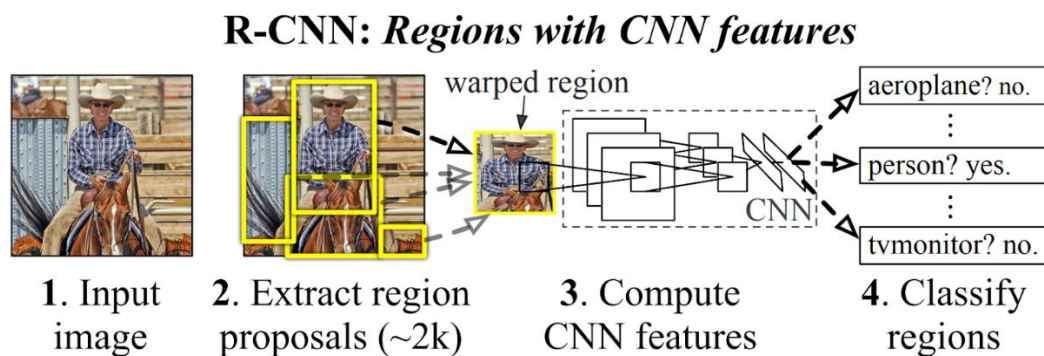
Thị giác máy tính(Computer Vision) là một trong những lĩnh vực quan trọng của Trí tuệ nhân tạo (Artificial Intelligence - AI). Thị giác máy tính là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý, phân tích và phân loại(image classification) các hình ảnh kỹ thuật số, định vị vật thể(object localization) và phát hiện các đối tượng trong hình ảnh hoặc đoạn phim. Và phát hiện đối tượng có lẽ là khía cạnh sâu sắc nhất của thị giác máy vì sự xuất hiện phổ biến trong các ứng dụng thực tế.

Một vài mô hình phát hiện đối tượng phổ biến

### 3.2.1. R-CNN

R-CNN được giới thiệu lần đầu vào năm 2014 bởi Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik thuộc UC Berkeley một trong những trung tâm nghiên cứu Trí tuệ nhân tạo hàng đầu thế giới. Trong bài báo Rich feature hierarchies for accurate object detection and semantic segmentation [2]

Đây là kiến trúc của mô hình được giới thiệu trong bài báo



Ảnh 32: Kiến trúc RCNN(Nguồn [2])



R-CNN gồm 3 thành phần chính:

- Đề xuất vùng hình ảnh(Region proposals): có nhiệm vụ đề xuất các vùng chứa đối tượng, ta có thể thấy số lượng hình ảnh đề xuất rất lớn (gần 2000)
- Trích xuất đặc trưng(Feature extraction): sử dụng một mạng nơron tích chập(CNN) để trích xuất các đặc trưng từ mỗi ảnh nhỏ chứa các đối tượng mà vùng đề xuất hình ảnh đưa ra. Đầu ra của mô-đun này là một vec-tơ 4096 chiều mô tả thông tin của hình ảnh
- Phân loại(Classify): từ các đặc trưng tính được ở phần trước mà phân loại hình ảnh, gán nhãn cho hình ảnh như bài toán phân loại hình ảnh(image classification)

Mô hình trên là một mô hình tương đối đơn giản và dễ hiểu với vấn đề phát hiện đối tượng, tuy nhiên điểm yếu của mô hình này là rất chậm vì phải qua lần lượt từng mô-đun trong đó việc đưa gần 2000 ảnh qua mô-đun trích xuất đặc trưng để tính toán thông tin mô tả hình ảnh là một điều khủng khiếp. Và đó cũng chính là vấn đề thiết yếu cần giải quyết để cải thiện tốc độ của mô hình này

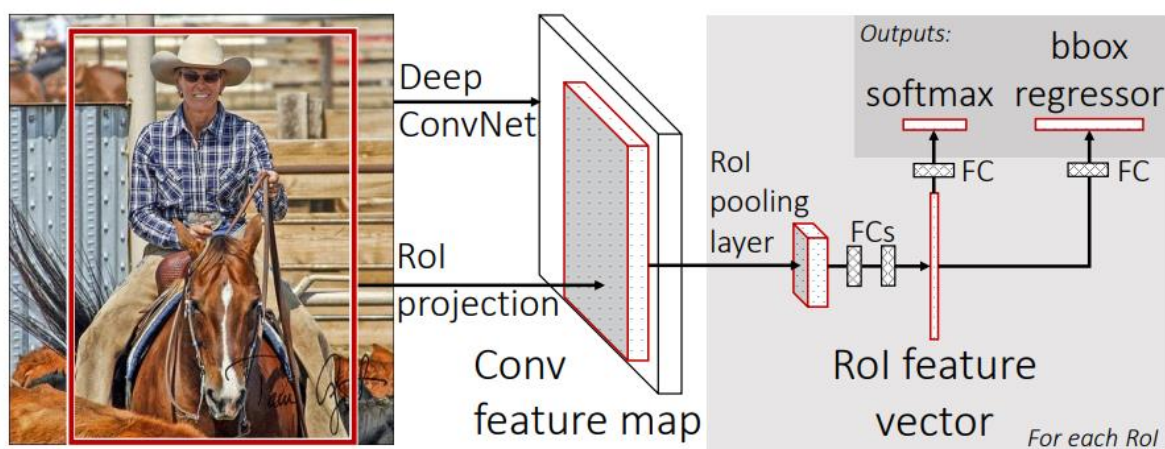
### **3.2.2. Fast R-CNN**

Năm 2015 sau khi chuyển sang Microsoft Research thì Ross Girshick đưa ra một giải pháp để giải quyết vấn đề của R-CNN trong bài báo Fast R-CNN [3]

Bài báo đã chỉ ra nhược điểm của R-CNN như sau:

- Huấn luyện (Training) qua một loạt các mô-đun riêng biệt, mỗi mô-đun lại vận hành nhưng mô hình(model) khác nhau
- Chi phí huấn luyện quá cao và phát hiện đối tượng chậm vì vấn đề đưa 2000 hình ảnh đề xuất vào mô-đun trích xuất đặc trưng để tính toán

Đây là kiến trúc mô hình bài báo đã đưa ra để giải quyết các vấn đề trên



Ảnh 33: Kiến trúc mô hình Fast R-CNN(Nguồn [3])

Điểm đặc biệt và đột phá của kiến trúc mới này là nó sử dụng chỉ một mô hình(model) duy nhất để phát hiện đối tượng và phân loại đối tượng cùng lúc.

Kiến trúc này hoạt động như sau:

- Từ ảnh đầu vào qua một mạng nơron tích chập sâu(Deep Convolutional Neural Network) kí hiệu là DeepConvNet như ảnh trên để tính toán ra bản đồ đặc trưng(Feature map).
- Cũng như R-CNN thì Fast R-CNN cũng có các vùng hình ảnh đề xuất (Region Proposals). Từ các vùng đề xuất này kiến trúc sử dụng một phép chiếu gọi là RoI projection để lấy được hình chiếu của vùng đề xuất ở trên bản đồ đặc trưng. Điểm cải tiến ở đây chính là thay vì tính toán đặc trưng cho 2000 vùng đề xuất ta chỉ tính toán đặc trưng đúng một lần cho ảnh đầu vào, sau đó lấy được đặc trưng của từng vùng đề xuất qua phép chiếu RoI projection
- Sau khi qua phép chiếu RoI projection thì với mỗi vùng đề xuất(Region Proposal) ta có được một tầng vùng quan tâm (Region Of Interest Pooling layer - RoI Pooling layer). Kết nối tầng này với các tầng kết nối đầy đủ(Fully Connected) ta có được một véc-tơ mô tả thông tin của đối tượng trong vùng đề xuất (được gọi là RoI Feature vector), véc-tơ này tương tự như trong mạng R-CNN

- Từ véc-tơ mô tả thông tin kiến trúc chia thành hai nhánh, một để dự đoán đối tượng và một để xác định toạ độ của hộp giới hạn (bounding box) của đối tượng

Kiến trúc trên đã giải quyết được vấn đề tốc độ huấn luyện cũng như tốc độ dự đoán với độ chính xác tương đương.

	Fast R-CNN			R-CNN			SPPnet
	<b>S</b>	<b>M</b>	<b>L</b>	<b>S</b>	<b>M</b>	<b>L</b>	<sup>†</sup> <b>L</b>
train time (h)	<b>1.2</b>	2.0	9.5	22	28	84	25
train speedup	<b>18.3×</b>	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	<b>0.06</b>	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	<b>213×</b>	-	-	-	-
VOC07 mAP	57.1	59.2	<b>66.9</b>	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

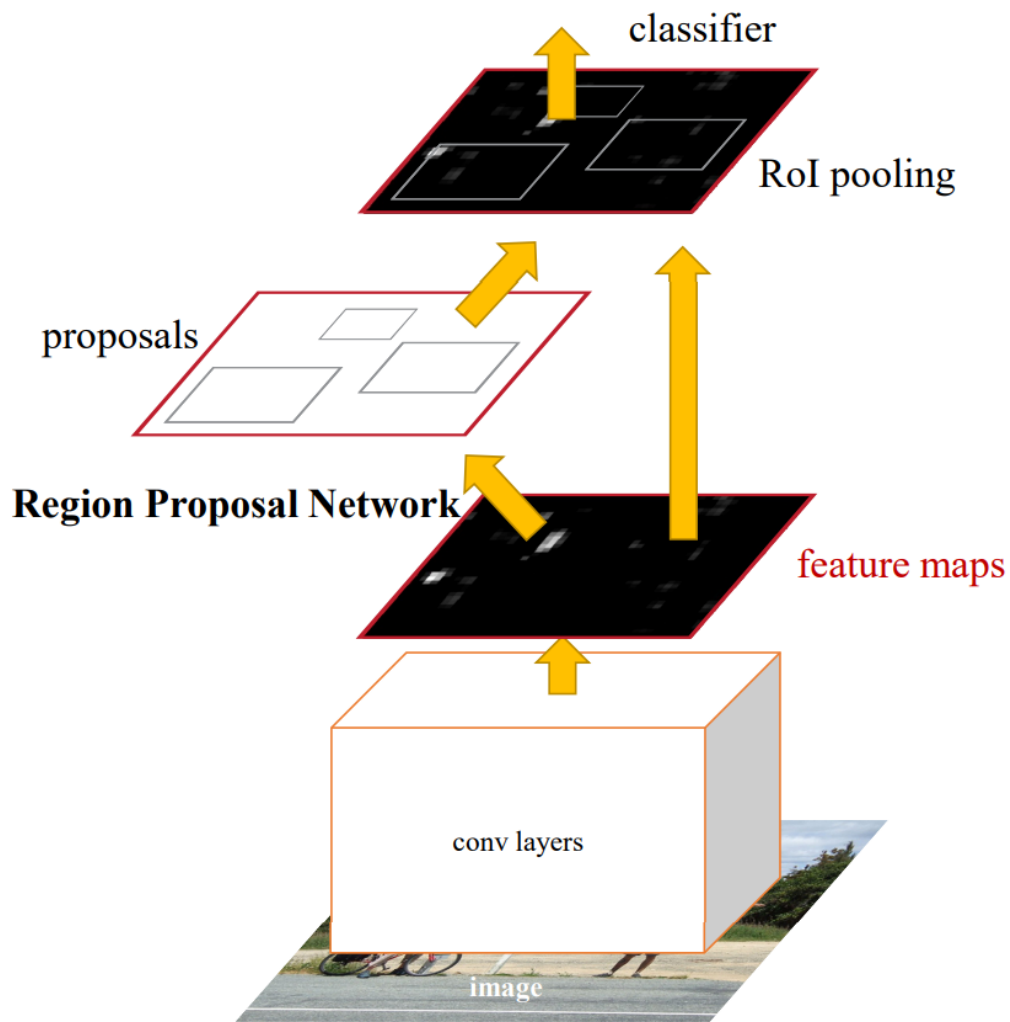
Ảnh 34: Bảng so sánh giữa Fast R-CNN và R-CNN trên cùng một GPU Nvidia K40(Nguồn [3])

### 3.2.3. Faster R-CNN

Năm 2016 Shaoqing Ren và các cộng sự tại Microsoft Research tiếp tục cải tiến kiến trúc phát hiện đối tượng trong bài báo Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [4]

Kiến trúc này vừa cải thiện được tốc độ cũng như độ chính xác và đã đạt được độ chính xác cao nhất trong cả hai tác vụ phát hiện và nhận dạng đối tượng trong hai cuộc thi ILSVRC-2015 và MS COCO-2015.

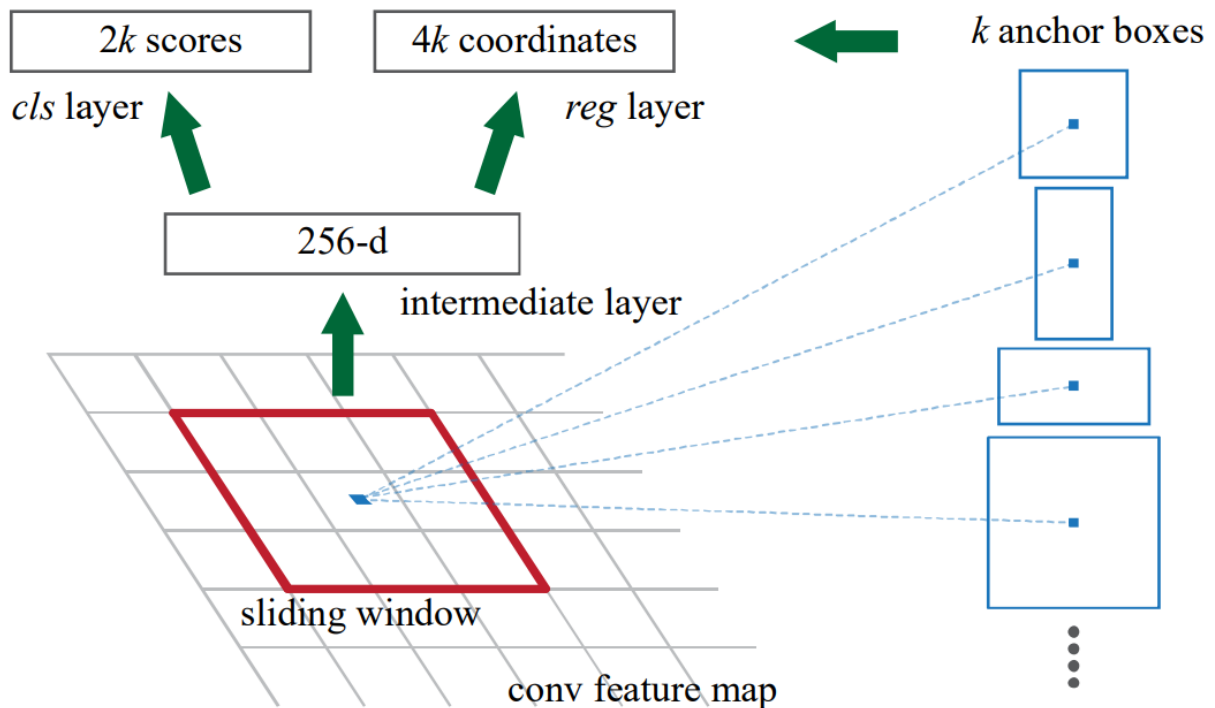
Kiến trúc của mô hình như sau



Ảnh 35: Mô hình Faster-RCNN(Nguồn [4])

Tương tự với Fast R-CNN thì kiến trúc này cũng có một mạng tích chập sâu để trích xuất ra một bản đồ đặc trưng, sau khi có được các vùng đề xuất thì kiến trúc cũng dùng phép chiếu RoI để lấy hình chiếu của các vùng đề xuất trên bản đồ đặc trưng rồi nhận diện các đối tượng trên đó. Điểm khác biệt của Faster R-CNN chính là việc thay vì dùng một thuật toán bên ngoài để đưa ra các vùng đề xuất thì kiến trúc này có riêng một mạng đề xuất(Region Proposal Networks - RPN), và mạng này hoạt động như một tiện ích mở rộng(extension) cho Faster R-CNN

Cách hoạt động của RPN như sau



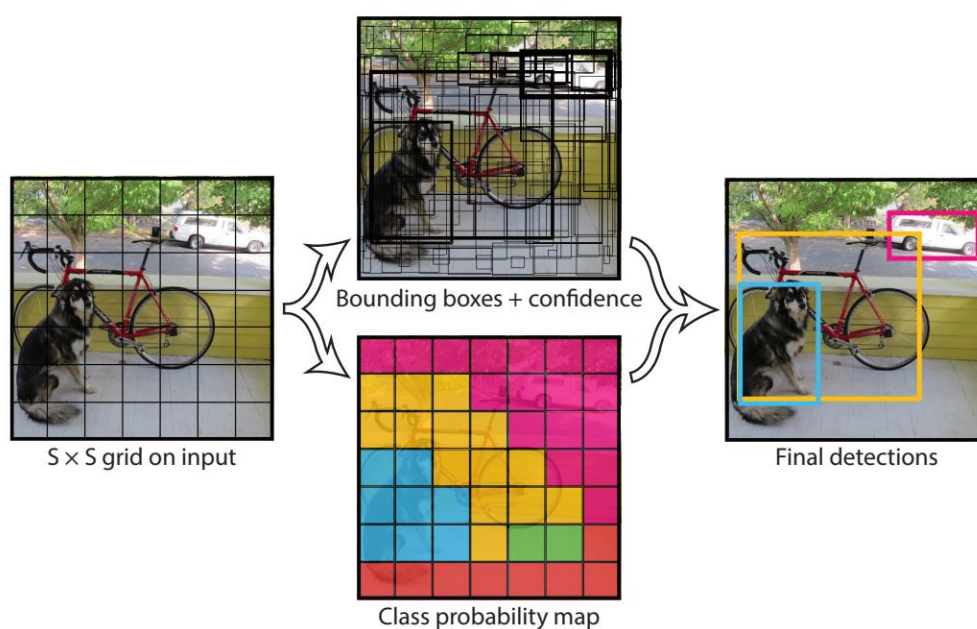
Ảnh 36: Cách hoạt động của RPN(Nguồn [4])

Để tạo ra các vùng đề xuất(region proposals) thì kiến trúc tạo ra một mạng gọi là các cửa sổ trượt(sliding-window) kích thước  $n \times n$  (trong bài báo gốc thì  $n=3$ ) trượt lên bản đồ đặc trưng thu được ở một mạng trích xuất đặc trưng. Mỗi cửa sổ trượt này RPN tạo ra K hộp neo có tâm trùng với tâm của cửa sổ trượt với 3 tỉ lệ dài rộng(aspect ratio) và 3 tỉ lệ so với ảnh gốc(scale) khác nhau, nên  $K=9$ . Đầu ra của mạng này là đầu vào của 2 lớp kết nối đầy đủ là CLS layer và REG layer. REG(box-regression)layer dự đoán vị trí của vùng đề xuất và CLS(box-classification) layer dự đoán xác suất chứa đối tượng của mỗi hộp đó.

### 3.2.4. YOLO, YOLOv2, YOLOv3

YOLO (You Only Look Once) được giới thiệu lần đầu tiên vào năm 2015 bởi Joseph Redmon và các cộng sự ở Facebook AI Research trong bài báo “You Only Look Once: Unified, Real-time Object Detection”

Phương pháp này sử dụng duy nhất một mạng nơ-ron(neural network). Mô hình nhận đầu vào là một ảnh và đầu ra là các hộp giới hạn(bounding box) và các nhãn đánh dấu đối tượng nào ở trong các hộp giới hạn đó. Vì không sử dụng vùng đề xuất(Region Proposal) nên độ chính xác thấp hơn(nhiều lỗi định vị đối tượng-localization loss) tuy nhiên đổi lại kiến trúc này hoạt động ở tốc độ cao hơn rất nhiều so với các kiến trúc R-CNN, Fast R-CNN hay Faster R-CNN. Cụ thể nó hoạt động ở tốc độ 45fps (frame per second - khung hình trên giây) ở phiên bản thường còn đối với bản tối ưu tốc độ là 155fps.



Ảnh 37: Mô tả cách thức hoạt động của mô hình YOLO(Nguồn [5])

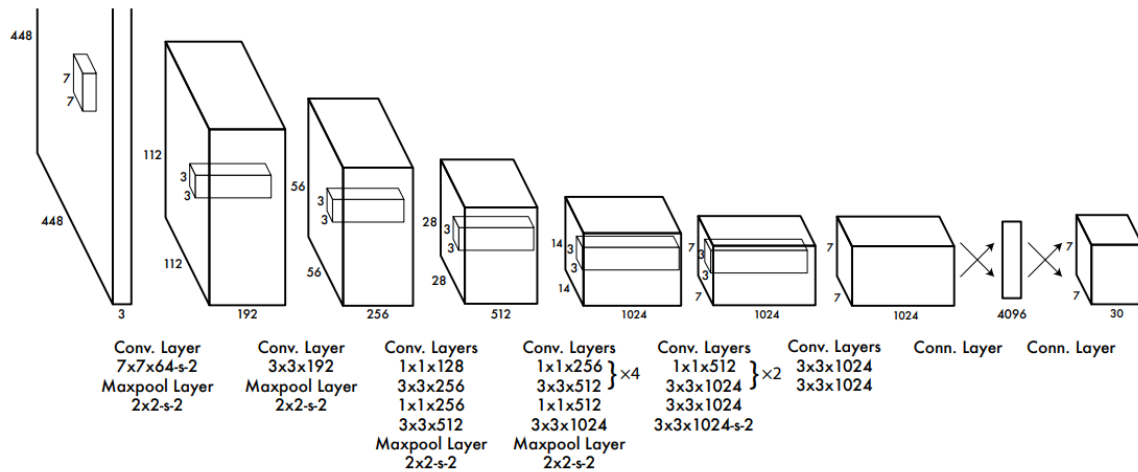
Mô hình hoạt động bằng cách chia ảnh đầu vào thành lưới các ô (grid of cells). Trên mỗi ô mô hình sẽ dự đoán hộp giới hạn(bounding box) dựa trên tọa độ x, y của tâm, chiều rộng(width), chiều cao(height). Ngoài ra nó còn dự đoán độ tin cậy(confidence) về khả năng có đối tượng bên trong cùng nhãn của đối tượng đó là gì. Sau đó, một sơ đồ xác suất nhãn (gọi là class probability map) với các confidence được kết hợp thành một tập hợp hộp giới hạn cuối cùng và các nhãn.

Theo ví dụ trên hình thì ảnh đầu vào được chia làm lưới  $S \times S$  ô, với mỗi ô ta dự đoán trên B hộp giới hạn, và độ tin cậy cho C lớp(classes - số loại các nhãn đối tượng). Như vậy

đầu ra sẽ là một tensor với kích thước  $S \times S \times (B \times 5 + C)$  trong đó 5 ở đây là  $x, y, w, h, c$  tương ứng với toạ độ tâm(gồm  $x, y$ ) chiều rộng(w), chiều dài(height), độ tin cậy(c).

Để đánh giá YOLO trên bộ dữ liệu PASCAL VOC thì tác giả sử dụng  $S=7, B=2$ .

PASCAL VOC thì có 20 nhãn ứng với 20 loại đối tượng khác nhau nên  $C=20$  thế nên đầu ra sẽ là một tensor với kích thước  $7 \times 7 \times 30$ .



Ảnh 38: Mô hình YOLO trên bộ dữ liệu PASCAL VOC(Nguồn [5])

Kiến trúc của mô hình trên gồm 24 lớp tích chập(Convolutional Layer) và theo sau là 2 lớp kết nối đầy đủ(Fully Connected)

Vào năm 2016 và 2018 thì có cải tiến thêm hai phiên bản YOLOv2 và YOLOv3 nâng số loại đối tượng nhận diện lên 9000 và cơ chế tính hộp giới hạn được tinh chỉnh để mô hình chạy ổn định hơn.

### 3.2.5. Single Shot MultiBox Detector(SSD)

SSD được giới thiệu vào cuối năm 2016 bởi Wei Liu cùng các cộng sự ở bài báo SSD : Single Shot MultiBox Detector(trích dẫn)

SSD cùng với các mô hình YOLO đang là những mô hình hiện đại nhất(state of the art) trong bài toán phát hiện đối tượng(Object Detection) nhờ tốc độ xử lý cao nhưng vẫn giữ được sự chính xác.

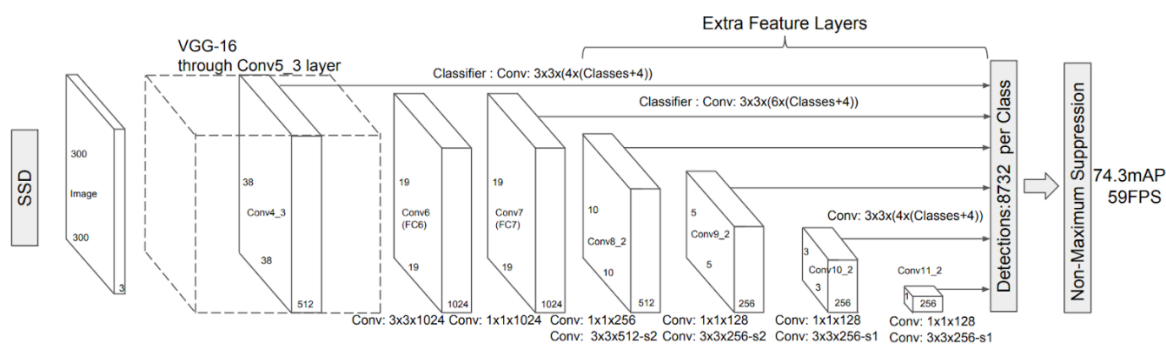
Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

*Ảnh 39: Ảnh trích từ bảng 7 trong bài báo gốc(Nguồn [6])*

Cách tiếp cận của mô hình SSD là nhận dạng các đối tượng trong một bản đồ đặc trưng(Feature Map), là đầu ra của một mạng nơ-ron tích chập sâu sau khi bỏ các tầng kết nối đầy đủ(Fully Connected Layer) ở cuối. Sau đó mô hình sẽ tạo ra một lưới các ô và với mỗi ô sẽ có một số lượng nhất định các hộp giới hạn có tâm là tâm của ô và kích thước và tỉ lệ dài rộng khác nhau dùng để dự đoán đối tượng có bên trong hay không. Kiến trúc cho phép nhận diện trên nhiều bản đồ đặc trưng với các độ phân giải khác nhau giúp nhận dạng được nhiều vật thể với các hình dạng và kích thước khác nhau và giống như YOLO thì kiến trúc này cũng không sử dụng vùng đề xuất(Region Proposals) mà tất cả quá trình phát hiện cũng như phân loại đối tượng đều trên một mạng duy nhất.

Dưới đây là kiến trúc của mô hình :

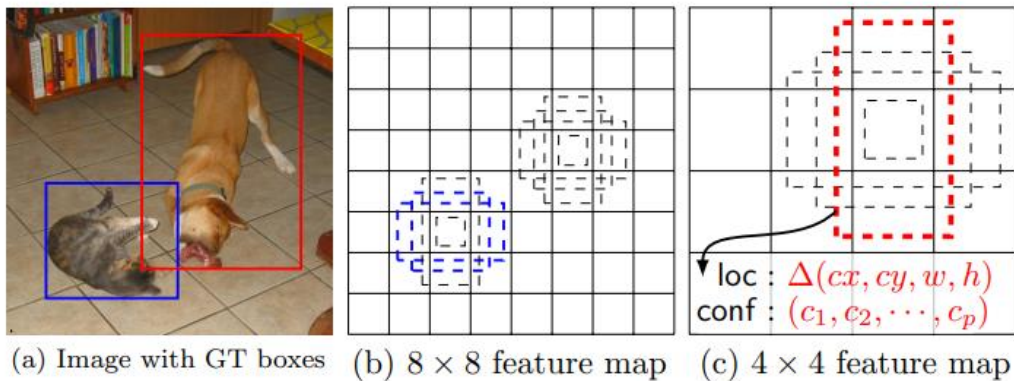




*Ảnh 40: Kiến trúc mô hình SSD(Nguồn [6])*

Đầu tiên ta thấy hình ảnh đầu vào là một ảnh kích thước 300x300 sau đó nó sẽ đi qua một mạng tích chập sâu để thu được một bản đồ đặc trưng(Feature map). Mạng tích chập sâu này được gọi là bộ trích xuất đặc trưng của mô hình SSD này. Có rất nhiều bộ trích xuất khác nhau như: VGG-16, MobileNet, Resnet fpn, ... Như ảnh trên thì mô hình hiện tại đang sử dụng bộ trích xuất VGG-16.

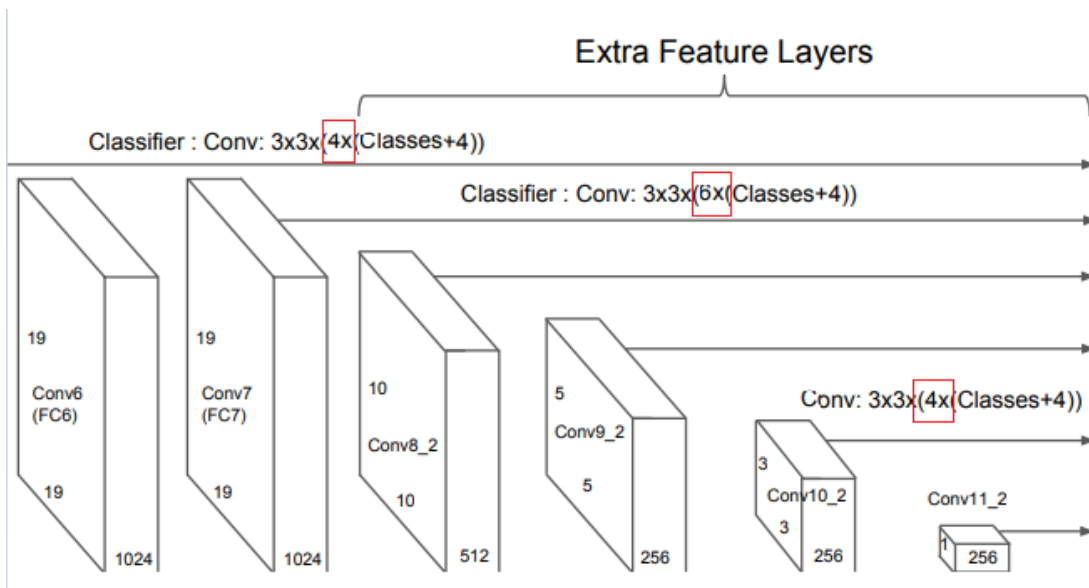
Sau đó mạng đi qua một loạt các lớp tích chập(Convolutional layer) để giảm dần kích thước của bản đồ đặc trưng xuống(việc đi qua các lớp tích chập không chỉ mang ý nghĩa giảm kích thước, mà còn tăng tính ngữ nghĩa cho bản đồ đặc trưng) đồng thời cũng bắt đầu dự đoán trên các bản đồ đặc trưng này. Nghĩa là kiến trúc cho phép dự đoán trên nhiều bản đồ đặc trưng với kích thước khác nhau, kết hợp với các hộp giới hạn ban đầu có kích thước và tỉ lệ dài rộng khác nhau nhằm nhận diện được những đối tượng có kích thước và tỉ lệ đa dạng. Cơ chế này được giải thích như thế này:



Ảnh 41: Mô tả hộp giới hạn với nhiều kích thước và tỉ lệ khác nhau(Nguồn [6])

Giả sử từ ảnh đầu vào là ảnh a, sau khi qua mạng trích xuất ta có được một bản đồ đặc trưng (Feature map) với kích thước  $8 \times 8$  thì mô hình sẽ chia thành một lưới  $8 \times 8$  còn khi kích thước của bản đồ đặc trưng là  $4 \times 4$  thì sẽ chia thành lưới  $4 \times 4$ , và như trên hình thì ta thấy là với mỗi ô thì ta sẽ có 4 hộp giới hạn mặc định, tâm của hộp giới hạn trùng với tâm của ô, chiều dài và chiều rộng của hộp sẽ theo 1 tỉ lệ nhất định đối với kích thước của một ô trên lưới. Như hình trên rõ ràng với các hộp đặc trưng mặc định như ở hình b (lưới  $8 \times 8$ ) mô hình không thể tìm được hộp giới hạn tốt chưa nguyên hình chú chó được, cũng như ở hình c (lưới  $4 \times 4$ ) thì mô hình cũng không thể nhận dạng được con mèo.

Giờ hãy nhìn lại hình ảnh kiến trúc mô hình ở trên đối với mỗi bản đồ đặc trưng thu được ở từng bước thì mô hình lại chia ra thành từng lưới và nhận dạng ở trên những lưới các ô đó.



Ảnh 42 Ảnh trích xuất từ bài báo gốc và thêm chú thích. (Nguồn [6])

Những ô màu đỏ chính là số hộp giới hạn mặc định trên từng lưới, từ đây ta có thể tính được tổng số hộp giới hạn mà mô hình dùng để dự đoán:

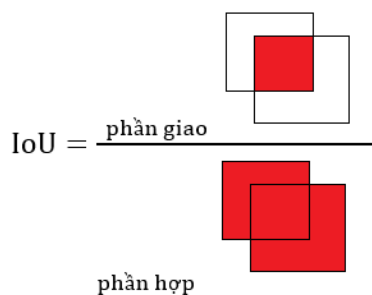
- Ở Conv4\_3 bản đồ đặc trưng 38x38x512 sử dụng 4 hộp giới hạn mặc định nên có  $38 \times 38 \times 4 = 5776$  hộp
- Ở Conv7  $19 \times 19 \times 6 = 2166$  hộp
- Conv8\_2  $10 \times 10 \times 6 = 600$  hộp
- Conv9\_2  $5 \times 5 \times 6 = 150$  hộp
- Conv10\_2  $3 \times 3 \times 4 = 36$  hộp
- Conv11\_2  $1 \times 1 \times 4 = 4$  hộp

Vậy tổng số hộp mà mô hình sẽ dự đoán là  $5776 + 2166 + 600 + 150 + 36 + 4 = 8732$  hộp, đúng như bảng 7 ở bài báo gốc

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Ảnh 43 Ảnh trích xuất từ bài báo gốc. Số hộp giới hạn dự đoán của mô hình SSD300(Nguồn [6])

Với đầu ra là 8732 hộp giới hạn thì đây là một con số quá lớn, nên kiến trúc cũng có giải pháp để xử lý vấn đề này. Đầu tiên thì sử dụng một ngưỡng 0.01 để loại tất cả những hộp có điểm tin cậy (confidence score) bé hơn ngưỡng này. Phương pháp này loại bỏ phần lớn các hộp không đáng tin cậy. Tiếp theo sử dụng thuật toán Non-Maximum-Suppression như sau. Thuật toán sẽ sắp xếp các dự đoán theo thứ tự điểm tin cậy. Sau đó bắt đầu từ hộp có điểm tin cậy cao nhất, tìm và loại tất cả các hộp có chỉ số IoU > 0.45 và cùng nhãn đối tượng với hộp đang xét, điều này nhằm loại bỏ các hộp khác nhau có dự đoán cùng một đối tượng. Chỉ số IoU (Intersection of Union) là chỉ số cho biết tỉ lệ giao nhau giữa hai khung hình và được tính theo công thức:

$$IoU = \frac{\text{phần giao}}{\text{phần hợp}}$$


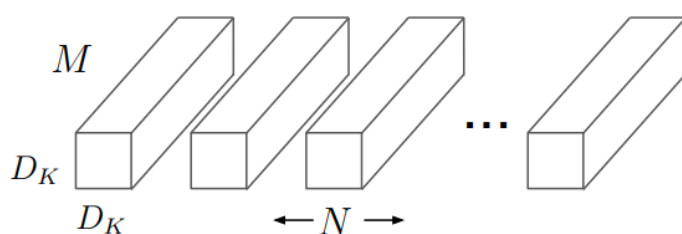
### 3.3. Bộ trích xuất trong các mô hình phát hiện đối tượng

#### 3.3.1. MobileNet

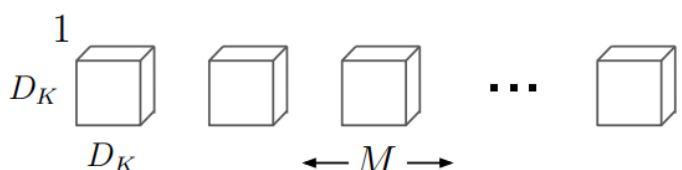
MobileNet được giới thiệu vào năm 2017 bởi Andrew G. Howard và các cộng sự ở Google Inc trong bài báo MobileNets: Efficient Convolutional Neural Networks for

Mobile Vision Applications[7]. Đây là một kiến trúc được ra đời với mục đích giảm số lượng tính toán tối đa để có được một mô hình nhẹ, phù hợp với các nền tảng bị giới hạn tính toán như điện thoại hay những thiết bị nhúng. Ta sẽ không bàn về kiến trúc của mô hình này mà sẽ bàn về những điểm đột phá giúp mô hình trở nên nhẹ nhất có thể nhưng vẫn giữ được độ chính xác cần thiết.

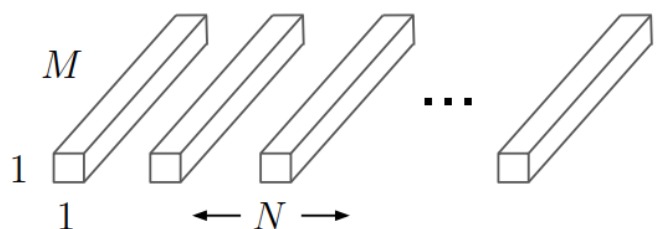
- Tích chập sâu tách rời(Depthwise Separable Convolution)



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

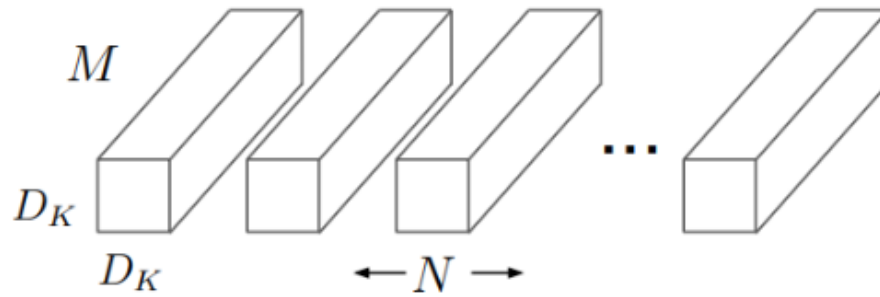


(c)  $1 \times 1$  Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Ảnh 44: Ảnh được trích từ bài báo gốc, cách chuyển một lớp tích chập thông thường thành lớp tích chập sâu tách rời(Nguồn [7])

Lớp tích chập sâu tách rời này bao gồm hai thành phần: Depthwise và Pointwise. Hình trên mô tả cách chuyển từ lớp tích chập thông thường sang tích chập sâu tách rời.

Đầu tiên ta xét đầu vào sẽ là một bản đồ đặc trưng (Feature map) kích thước  $D_f \times D_f \times M$  với  $D_f$  là chiều dài và rộng,  $M$  là số kênh (channel). Đầu ra sẽ là một bản đồ đặc trưng khác với kích thước  $D_f \times D_f \times N$  với  $D_f$  là chiều dài và rộng. Để thu được đầu ra như vậy lớp tích chập thông thường cần có  $N$  bộ lọc (filter) kích thước  $D_k \times D_k \times M$ . Đó chính là thứ được mô tả trong hình này



(a) Standard Convolution Filters

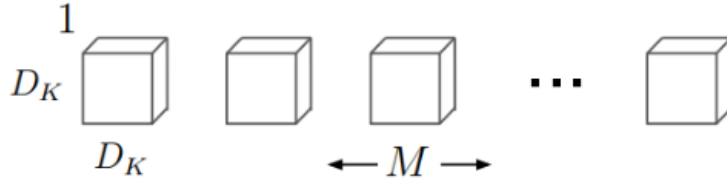
Ảnh 45 Ảnh trích xuất từ bài báo gốc, bộ lọc của lớp tích chập thông thường (Nguồn [7])

Chi phí tính toán cho việc áp dụng lớp này vào để có được đầu ra từ đầu vào như mô tả ở trên là  $D_f \times D_f \times M \times N \times D_k \times D_k$  (1)

Bây giờ chúng ta sẽ thay đổi lớp trên thành lớp tích chập sâu tách rời. Thay vì sử dụng nguyên bản đồ đặc trưng kích thước  $D_f \times D_f \times M$  với bộ lọc (filter)

$D_k \times D_k \times M$  thì lớp mới tách ra đầu vào  $D_f \times D_f \times M$  thành  $M$  kênh (channel)

khác nhau. Sau đó áp dụng bộ lọc (filter) kích thước  $D_k \times D_k \times 1$  với từng kênh của đầu vào, đây chính thành phần Depthwise

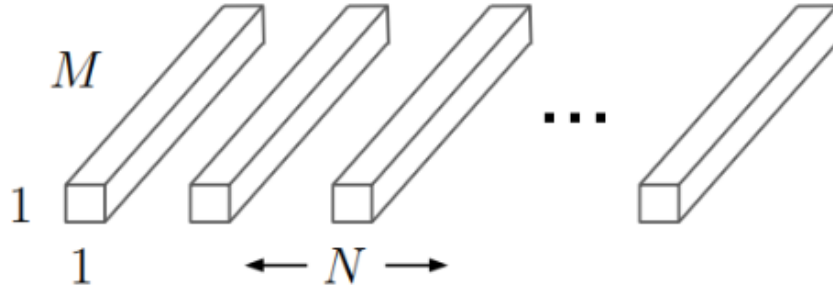


(b) Depthwise Convolutional Filters

Ảnh 46 Ảnh trích xuất từ bài báo gốc, bộ lọc Depthwise(Nguồn [7])

Sau khi qua bước này đầu ra sẽ là bản đồ đặc trưng mới với kích thước  $D_f \times D_f \times M$  và chi phí tính toán cho bước này là  $D_f \times D_f \times M \times D_k \times D_k$

Tiếp theo ta sẽ tới với phần Pointwise. Ở đây ta sử dụng N bộ lọc có kích thước  $1 \times 1 \times M$



Ảnh 47 Ảnh trích xuất từ bài báo gốc, bộ lọc Pointwise(Nguồn [7])

Với đầu vào là đầu ra của Depthwise  $D_f \times D_f \times M$  thì qua phần này đầu ra sẽ là  $D_f \times D_f \times N$ , đúng với yêu cầu ban đầu, và chi phí tính toán cho bước này là:  $D_f \times D_f \times M \times N$

Vậy tổng chi phí ở Depthwise Separable Convolution sẽ là

$$D_f \times D_f \times M (N + D_k \times D_f) \quad (2)$$

Tỉ lệ của chi phí (2) so với chi phí (1) là :

$$\frac{D_f * D_f * M(N + D_k * D_k)}{D_f * D_f * M * N * D_k * D_k} = \frac{1}{D_k * D_k} + \frac{1}{N}$$

Và đây là kết quả

**Table 4. Depthwise Separable vs Full Convolution MobileNet**

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

*Ảnh 48 Ảnh trích xuất từ bài báo gốc, so sánh chi phí tính toán cũng như số lượng tham số của mô hình sử dụng tích chập thông thường và tích chập sâu tách rời. (Nguồn [7])*

Có thể thấy là chi phí chỉ giảm 88% , số lượng tham số giảm 85% nhưng sự chính xác thì giảm không đáng kể.

- Hai tham số Width Multiplier  $\alpha$  và Resolution Multiplier  $\rho$ .
  - Tham số  $\alpha$  điều chỉnh số lượng channel(M và N), chi phí tính toán của tích chập sâu tách rời với tham số  $\alpha$  sẽ là:

$$D_f * D_f * \alpha M * D_k * D_k + D_f * D_f * \alpha M * \alpha N$$

Giá trị  $\alpha$  sẽ nằm trong khoảng [0,1] với bước nhảy là 0.25 và các giá trị của  $\alpha$  sẽ là 0.25, 0.5, 0.75 còn nếu là 1 thì chính là mạng MobileNet ban đầu của mình. Giá trị  $\alpha$  càng bé thì chi phí tính toán cũng như số lượng tham số của mô hình sẽ giảm và tất nhiên độ chính xác cũng sẽ giảm đáng kể. Dưới



đây là bảng so sánh được lấy từ bài báo gốc

**Table 6. MobileNet Width Multiplier**

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

*Ảnh 49 Ảnh trích xuất từ bài báo gốc, so sánh số phép tính toán, số lượng tham số và độ chính xác của mô hình với các giá trị  $\alpha$  (Nguồn [7])*

- Tham số  $\rho$  điều chỉnh độ phân giải của ảnh đầu vào, chi phí tính toán khi sử dụng thêm tham số  $\rho$  là :

$$\rho D_f * \rho D_f * \alpha M * D_k * D_k + \rho D_f * \rho D_f * \alpha M * \alpha N$$

Giá trị của  $\rho$  cũng nằm trong khoảng [0,1] nhóm tác giả sử dụng các độ phân giải 224, 192, 160, và 128 và với chỉ số  $\rho$  giảm thì số phép tính giảm nhưng số lượng tham số của mô hình thì không, độ chính xác của mô hình thì cũng sẽ giảm vì rõ ràng độ phân giải giảm thì sẽ khiến mất đi một số đặc trưng của ảnh. Dưới đây là bảng so sánh được lấy từ bài báo gốc

**Table 7. MobileNet Resolution**

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

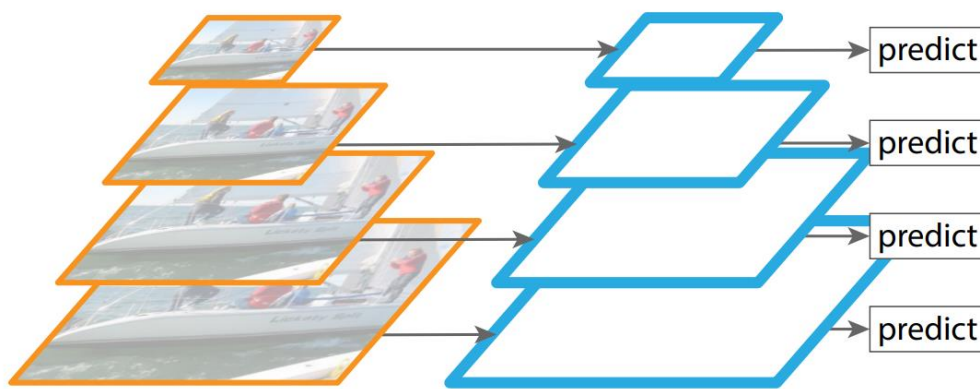
*Ảnh 50 Ảnh trích xuất từ bài báo gốc, bảng so sánh số phép tính toán, số lượng tham số và độ chính xác của mô hình với các độ phân giải của ảnh đầu vào (Nguồn [7])*

### 3.3.2. ResnetFpn

- Mạng kim tự tháp đặc trưng (Feature Pyramid Networks)

Kiến trúc này được giới thiệu năm 2017 trong bài báo Feature Pyramid Networks For Object Detection (trích dẫn). Bài báo ra đời trong hoàn cảnh phát hiện đối tượng (Object Detection) với nhiều kích thước khác nhau (different scales) là đang một thách thức đối với Thị giác máy tính (Computer Vision).

Kiến trúc kim tự tháp đặc trưng xây dựng trên kim tự tháp hình ảnh (Feature pyramids built upon image pyramids - Featurized image pyramid) là một kiến trúc cơ bản đã giải quyết vấn đề này.



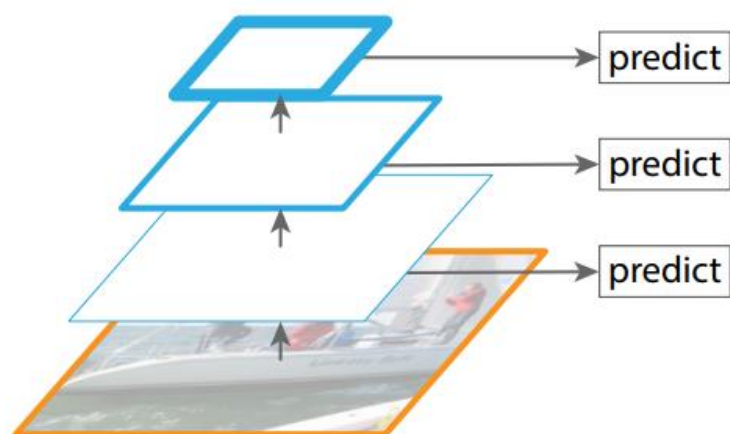
(a) Featurized image pyramid

*Ảnh 51: Ảnh từ bài báo gốc kiến trúc kim tự tháp đặc trưng xây dựng trên kim tự tháp hình ảnh (Nguồn [8])*

Ý tưởng của kiến trúc này là đưa một loạt ảnh với độ phân giải từ thấp đến cao (nhìn giống như kim tự tháp) qua một mạng tích chập sâu để trích xuất đặc trưng, đối với mỗi ảnh đưa vào ta sẽ có được một bản đồ đặc trưng (Feature map) từ đó có được một kim tự tháp đặc trưng từ các bản đồ đặc trưng có độ phân giải từ thấp đến cao như hình trên. Tuy nhiên điểm yếu của kiến trúc này là việc đưa hình ảnh ở nhiều kích thước khác nhau vào

mạng tích chập sâu để tính toán một cách độc lập thì sẽ rất chậm, đây là vấn đề giống với R-CNN ở thời điểm mới ra mắt.

Để khắc phục nhược điểm này thì một kiến trúc mới sử dụng một mạng tích chập sâu để tạo một bản đồ đặc trưng phân cấp(Pyramidal feature hierarchy) từ một hình ảnh duy nhất



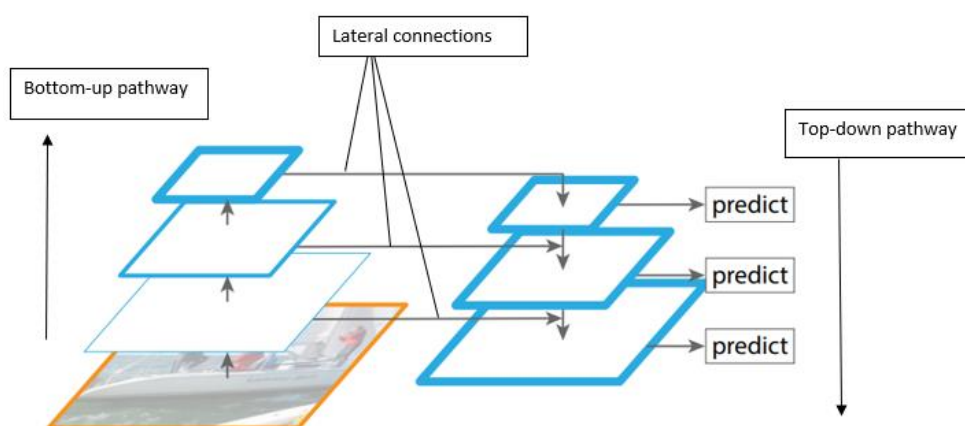
### (c) Pyramidal feature hierarchy

*Ảnh 52: Kiến trúc bản đồ đặc trưng phân cấp(Nguồn [8])*

Kiến trúc này đã được SSD sử dụng rất tốt và có độ chính xác cao trong việc phát hiện đối tượng. Giải thích một chút về những đường viền màu xanh có độ dày khác nhau trên các bản đồ đặc trưng(Feature map) thì càng qua nhiều lớp tích chập thì mức độ ngữ nghĩa(semantic) của bản đồ tính năng càng cao, đường viền dày thể hiện cho mức độ ngữ nghĩa cao tuy nhiên khi qua nhiều lớp tích chập thì độ phân giải của bản đồ đặc trưng cũng giảm xuống. Bản đồ đặc trưng có mức độ ngữ nghĩa càng cao thì càng tốt cho việc dự đoán các đối tượng của mô hình. Để tránh sử dụng những bản đồ đặc trưng có mức độ ngữ nghĩa thấp thì SSD đã sử dụng lại những mạng trích xuất tích năng như VGG-16 để lấy ra được một bản đồ tích năng có mức ngữ nghĩa đủ cao, sau đó mới thêm một số lớp tích chập để tạo thêm những bản đồ tính năng mới với mức ngữ nghĩa cao hơn cùng độ phân giải thấp hơn. Như hình ảnh trên thì ta có thể hình dung là từ ảnh ban đầu lên tới bản đồ đặc trưng đầu tiên thì SSD đã sử dụng một mạng trích xuất để lấy được nó việc này đã khiến cho những bản đồ đặc trưng có độ phân giải cao hơn, mức độ ngữ nghĩa

thấp hơn bị bỏ qua. Điều đó khiến cho nó mất đi cơ hội sử dụng những bản đồ đặc trưng có độ phân giải cao, thứ rất cần thiết trong việc nhận diện được các đối tượng có kích thước nhỏ.

Mục tiêu của bài báo này là xây dựng một kim tự tháp tính năng (Feature Pyramid) có mức độ ngữ nghĩa cao ở tất cả tầng của kim tự tháp. Và dĩ nhiên phải xây dựng nó từ một ảnh duy nhất để không làm cho tốc độ của kiến trúc quá chậm giống như kiến trúc đầu tiên. Để đạt được mục đích này thì kiến trúc dựa trên việc kết hợp những bản đồ đặc trưng có mức ngữ nghĩa cao, độ phân giải thấp với những bản đồ đặc trưng có mức ngữ nghĩa thấp, độ phân giải cao thông qua một đường đi từ trên xuống của kim tự tháp tính năng (top-down pathway) và kết nối bên (lateral connections).



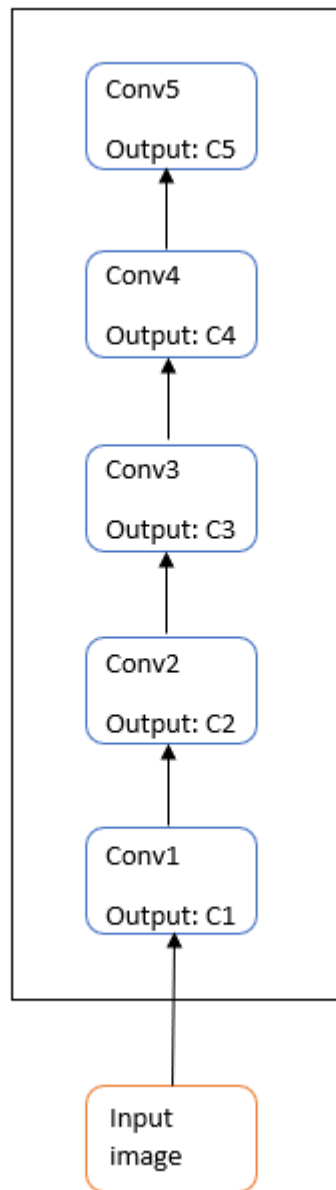
(d) Feature Pyramid Network

Ảnh 53: Ảnh được trích xuất từ bài báo gốc và thêm một số chú thích (Nguồn [8])

Kiến trúc bao gồm hai thành phần chính là hai đường từ dưới lên trên (Bottom-up pathway) và trên xuống dưới (Top-down pathway).

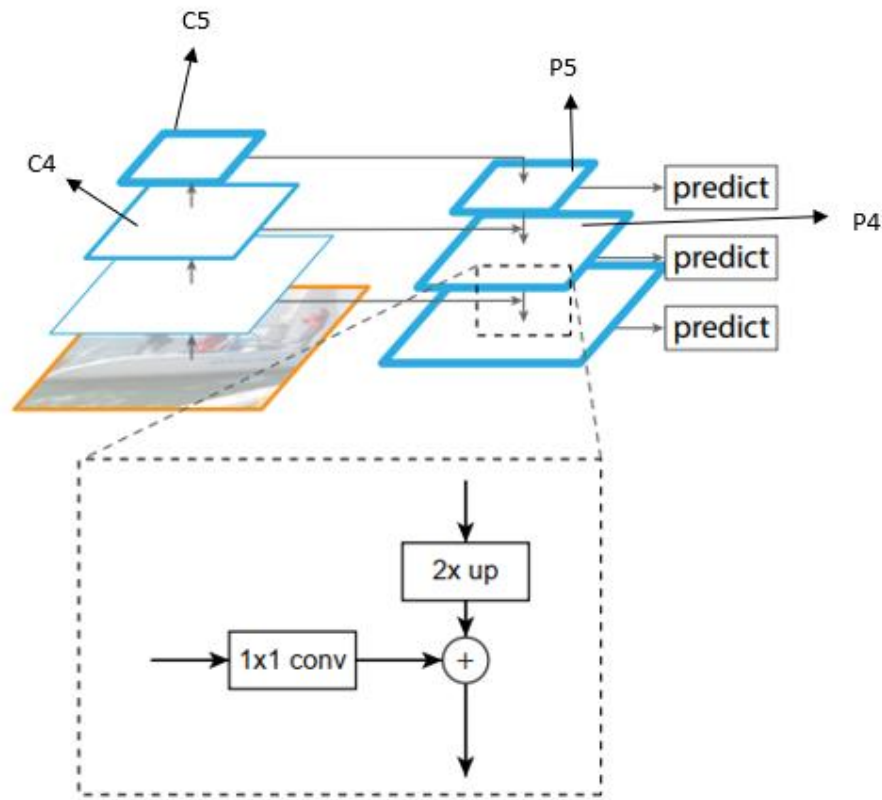
- Bottom-up pathway là một mạng tích chập sâu với mục đích trích xuất đặc trưng. Khi đi từ dưới lên (có nghĩa là đi qua các lớp tích chập) thì độ phân giải của bản đồ đặc trưng sẽ giảm và mức ngữ nghĩa sẽ tăng lên. Mạng tích chập này sẽ có nhiều tầng (layer) tính toán ra các bản đồ đặc trưng có cùng kích thước và tác giả gọi đó là những tầng có cùng cấp (stage) trong mạng. Tác giả cũng định nghĩa rằng đầu ra

của tầng cuối cùng trong một cấp(stage) chính là một bản đồ đặc trưng(Feature map) trong kim tự tháp đặc trưng, thứ mà sẽ được làm tăng tính ngữ nghĩa để tạo một kim tự tháp đặc trưng có tất cả các tầng của kim tự tháp đều là một bản đồ tính năng giàu tính ngữ nghĩa với bất kì độ phân giải nào. Hình dưới là mô tả của Bottom-up pathway với conv1 đến conv5 là các cấp của mạng, mỗi cấp sẽ cho ra một bản đồ đặc trưng được kí hiệu là C1 đến C5. Tuy nhiên C1 sẽ không được sử dụng vì kích thước lớn gây tốn bộ nhớ



*Ảnh 54 Mô tả Bottom-up pathway.*

- Top-down pathway



Ảnh 55: Ảnh lấy từ bài báo gốc và thêm mô tả(Nguồn [8])

Bắt đầu từ C5, tất cả các bản đồ đặc trưng trong tập  $\{C2, C3, C4, C5\}$  đều được qua một lớp tích chập  $1 \times 1$  để giảm số kênh(channel). Kết quả ta sẽ được tập hợp  $\{S2, S3, S4, S5\}$ . Bây giờ chúng ta sẽ xây dựng kim tự tháp tính năng của chúng ta, kết quả sẽ là tập  $\{P2, P3, P4, P5\}$ . Như hình trên thì ta thấy P5 chính là S5, và từ P5 muốn có được P4 thì ta sẽ upsample độ phân giải của P5 lên 2 lần bằng thuật toán nearest neighbor upsampling sau đó thực hiện phép cộng với S4 đã có ở trên. Tương tự như vậy ta tính được P3, P2.

Kiến trúc FPN không phải là một mạng để phát hiện đối tượng, nó chỉ là một dạng của mạng trích xuất để trích xuất ra bản đồ đặc trưng rồi các mạng như Faster R-CNN hay SSD sử dụng để phát hiện đối tượng. Không giống như những mạng trích xuất thông thường chỉ trả về một bản đồ đặc trưng duy nhất FPN trả về một loạt bản đồ đặc trưng có

độ phân giải từ thấp tới cao giống như một kim tự tháp và điểm đặc biệt là tất cả các bản đồ đặc trưng này đều có mức ngữ nghĩa cao để thuận lợi cho việc phát hiện đối tượng.

### **3.4. Mô hình nhận dạng ký tự quang học**

Đối với các ngôn ngữ phổ biến như tiếng Anh, hiện nay đã có nhiều mô hình nhận diện cho kết quả chính xác cao. Tuy nhiên đối với tiếng Việt, là ngôn ngữ có dấu, đồng thời bộ dữ liệu để huấn luyện chưa có nhiều. Chính vì vậy việc huấn luyện một mô hình nhận diện chữ tiếng Việt là một thử thách không nhỏ.

Do việc huấn luyện một mô hình nhận dạng tiếng Việt đòi hỏi một bộ dữ liệu lớn đa dạng cũng như quá trình huấn luyện lâu. Nhóm lựa chọn những mô hình đã được huấn luyện sẵn và cải tiến độ chính xác. Khác với một số loại giấy tờ định danh khác, thẻ căn cước công dân có nền màu trắng và chữ màu đen. Riêng phần số định danh có màu đỏ. Qua một vài kỹ thuật xử lý ảnh, hình ảnh thẻ căn cước có thể biến thành hình nhị phân với hai màu đen và trắng. Là đầu vào lý tưởng cho mô hình Tesseract, mô hình nhận diện chữ từ hình ảnh văn bản

Tesseract là một bộ thư viện mã nguồn mở được phát triển bởi Google. Thư viện này hỗ trợ trích xuất ký tự từ tập tin hình ảnh. Tesseract có khả năng nhận diện ký tự của hơn 100 ngôn ngữ khác nhau. Ngoài ra, đối với các font chữ chưa được huấn luyện, các ngôn ngữ mới, các hình ảnh chưa cho độ chính xác cao, thư viện này cho phép người sử dụng huấn luyện lại mô hình để cải thiện độ chính xác. Hạn chế của công cụ này là với các hình ảnh chưa được tiền xử lý, độ chính xác không cao.

### **3.5. Giải pháp nhóm chọn cho tác vụ phát hiện đối tượng**

Vì các đối tượng trên thẻ căn cước đều thuộc dạng đối tượng nhỏ nên nhóm quyết định sử dụng mô hình SSD cùng với bộ trích xuất đặc trưng ResnetFpn cho hai tác vụ nhận diện 4 góc của căn cước cũng như nhận diện các vị trí các thông tin trên thẻ căn cước. Mô hình này có khả năng nhận diện rất tốt các đối tượng nhỏ và nhận diện ở tốc độ cao



# Chương 4 Cài đặt giải pháp

## 4.1. Giới thiệu python và Tensorflow

### 4.1.1. Python

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao được phát triển bởi Guido Van Rossum. Đây là ngôn ngữ được nhiều người ưa chuộng và đặc biệt là với những người mới làm quen với lập trình. Python được sử dụng với rất nhiều mục đích như lập trình ứng dụng web, khoa học và tính toán, và được sử dụng rộng rãi để dạy lập trình. Lý do Python được ưa chuộng là bởi vì cách tiếp cận đơn giản, cú pháp rõ ràng, dễ hiểu. Ngoài ra Python còn có những ưu điểm sau:

- Miễn phí, mã nguồn mở: bạn có thể thoải mái sử dụng Python ngay cả cho mục đích thương mại. Vì là mã nguồn mở bạn không chỉ sử dụng nó mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng lớn không ngừng cải thiện nó theo thời gian.
- Hỗ trợ đa nền tảng: chương trình Python có thể chuyển từ nền tảng này sang nền tảng khác và chạy mà không cần sự thay đổi nào. Nó hỗ trợ trên tất cả các nền tảng hiện nay như MacOS, Linux hay Windows
- Nhiều thư viện hỗ trợ: Python có rất nhiều thư viện xử lý các tác vụ giúp cho công việc của các nhà phát triển(developer) trở nên dễ dàng và đặc biệt là các thư viện xử lý toán học đa dạng và mạnh mẽ. Các thư viện hỗ trợ máy học(Machine Learning) trong Python cũng rất nhiều và mạnh mẽ

### 4.1.2. Thư viện Tensorflow:

Tensorflow là một thư viện mã nguồn mở phục vụ tính toán số học sử dụng đồ thị luồng dữ liệu. Trong đó các nút(node) là các phép tính toán còn các cạnh là các luồng dữ liệu. Trong Tensorflow có một số khái niệm cơ bản như sau:

- Tensor: cấu trúc dữ liệu đại diện cho tất cả dữ liệu trong Tensorflow, một tensor có 3 thuộc tính cơ bản là:
- Bậc(Rank): số bậc của tensor(Scalar, vector, matrix, N-tensor):

Scalar: khi tensor có bậc là 0, ví dụ : số 2

Vector: khi tensor có bậc 1, ví dụ [1,2,3,4]

Matrix: khi tensor có bậc 2, ví dụ [[1,2],[3,4]]

N-tensor: khi tensor có bậc lớn hơn 2, ví dụ:

[[[1,2],[3,4]],[[5,6],[7,8]]]

- Số chiều(Shape): xác định số chiều của tensor ví dụ: [[1,2],[3,4]] có shape=(2,2) , [[1,2,3],[4,5,6]] có shape=(2,3)
- Kiểu dữ liệu(Type): kiểu dữ liệu của toàn bộ thành phần trong tensor
- Graph: là đồ thị với các nút(node) đại diện cho biến đầu vào hoặc phép tính toán, cạnh(edge) đại diện cho dữ liệu . Các dữ liệu trong tensorflow đều phải ở dạng tensor. Các đồ thị này mô tả cách tính toán chứ chưa chạy. Muốn chạy một graph ta cần một session
- Session: là môi trường để Tensorflow thực thi, nó như là một phiên xử lý. Mỗi phiên xử lý này có quyền sử dụng phần cứng và dựa vào các đồ thị graph để tính toán giá trị đầu ra

Đối với tác vụ phát hiện đối tượng thì trong Tensorflow có các mô hình để đào tạo, ngoài ra còn có những mô hình đã đào tạo(pretrained model) cũng như các tệp cấu hình(config file) đi theo để học chuyển(transfer learning).

## TensorFlow 2 Detection Model Zoo

TensorFlow 2.2 Python 3.6

We provide a collection of detection models pre-trained on the [COCO 2017 dataset](#). These models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. You can try it in our inference [colab](#).

They are also useful for initializing your models when training on novel datasets. You can try this out on our few-shot training [colab](#).

Finally, if you would like to train these models from scratch, you can find the model configs in this [directory](#) (also in the linked [tar.gz](#) s).

Model name	Speed (ms)	COCO mAP	Outputs
<a href="#">CenterNet HourGlass104 512x512</a>	70	41.6	Boxes
<a href="#">CenterNet HourGlass104 Keypoints 512x512</a>	76	40.0/61.4	Boxes/Keypoints
<a href="#">CenterNet HourGlass104 1024x1024</a>	197	43.5	Boxes
<a href="#">CenterNet HourGlass104 Keypoints 1024x1024</a>	211	42.8/64.5	Boxes/Keypoints
<a href="#">CenterNet Resnet50 V1 FPN 512x512</a>	27	31.2	Boxes
<a href="#">CenterNet Resnet50 V1 FPN Keypoints 512x512</a>	30	29.3/50.7	Boxes/Keypoints
<a href="#">CenterNet Resnet101 V1 FPN 512x512</a>	34	34.2	Boxes
<a href="#">CenterNet Resnet50 V2 512x512</a>	27	29.5	Boxes
<a href="#">CenterNet Resnet50 V2 Keypoints 512x512</a>	30	27.6/48.2	Boxes/Keypoints
<a href="#">EfficientDet D0 512x512</a>	39	33.6	Boxes
<a href="#">EfficientDet D1 640x640</a>	54	38.4	Boxes
<a href="#">EfficientDet D2 768x768</a>	67	41.8	Boxes
<a href="#">EfficientDet D3 896x896</a>	95	45.4	Boxes
<a href="#">EfficientDet D4 1024x1024</a>	133	48.5	Boxes
<a href="#">EfficientDet D5 1280x1280</a>	222	49.7	Boxes
<a href="#">EfficientDet D6 1280x1280</a>	268	50.5	Boxes
<a href="#">EfficientDet D7 1536x1536</a>	325	51.2	Boxes
<a href="#">SSD MobileNet v2 320x320</a>	19	20.2	Boxes
<a href="#">SSD MobileNet V1 FPN 640x640</a>	48	29.1	Boxes
<a href="#">SSD MobileNet V2 FPNLite 320x320</a>	22	22.2	Boxes
<a href="#">SSD MobileNet V2 FPNLite 640x640</a>	39	28.2	Boxes
<a href="#">SSD ResNet50 V1 FPN 640x640 (RetinaNet50)</a>	46	34.3	Boxes

Ảnh 56: Ảnh được lấy từ github của Tensorflow. Đây là các mô hình đã được dạy học trên bộ dữ liệu COCO2017

### 4.2. Cài đặt hai tác vụ cắt ảnh và nhận diện vị trí các thông tin trên thẻ căn cước

Mục tiêu của hai bước là từ ảnh đầu vào xác định tọa độ 4 góc của thẻ căn cước công dân, từ đó sử dụng các phép biến đổi toán học và kỹ thuật xử lý ảnh để cắt ảnh và xoay ảnh về trạng thái không bị nghiêng. Sau đó nhận diện vị trí các thông tin trên thẻ căn cước để phục vụ cho bước đọc thông tin

Cả hai tác vụ này cùng dựa vào bài toán nhận dạng đối tượng đã được xây dựng và giải quyết trong thực tế. Ta lần lượt xây dựng các bước để cài đặt và giải quyết bài toán này.

- Chuẩn bị dữ liệu, đa dạng hóa dữ liệu, loại bỏ các dữ liệu gây nhiễu để huấn luyện mô hình
- Huấn luyện mô hình

Chi tiết các bước cài đặt

#### **4.2.1. Chuẩn bị dữ liệu**

##### **4.2.1.1. Chuẩn bị ảnh**

Do thẻ căn cước công dân là loại giấy tờ cá nhân, mang tính riêng tư nên việc thu thập số lượng dữ liệu đủ lớn để việc huấn luyện mô hình mang lại hiệu quả cao là một công việc khó khăn. Từ các nguồn dữ liệu từ internet, nhóm thu thập được hơn 300 mẫu căn cước công dân. Trong đó có hơn 100 mẫu là các hình ảnh có 4 góc của thẻ căn cước. Để có đủ số lượng dữ liệu cần thiết, nhóm sử dụng một số kỹ thuật xử lý ảnh để tự động phát sinh một số hình ảnh có chứa thẻ căn cước công dân, mang các đặc điểm tương tự thẻ căn cước thật. Kết hợp với các thẻ căn cước thu thập được từ internet tạo nên bộ dữ liệu thống nhất.

Quá trình tạo ảnh giả trải qua nhiều bước. Đầu tiên là từ ảnh thật, ta cắt ra một thẻ căn cước ra khỏi nền ảnh rồi dùng Photoshop để xóa hết thông tin trên thẻ. Tiếp theo ta sẽ sử dụng những thuật toán để khởi tạo các thông tin trên thẻ một cách ngẫu nhiên rồi vẽ lên trên các ảnh đó để trông như một thẻ căn cước thật. Các hình ảnh này đang ở trạng thái bình thường, không bị nghiêng, không có nền và các chi tiết không liên quan. Sau đó, qua một số hàm được thư viện OpenCV hỗ trợ, đưa thẻ căn cước vào một ảnh nền ngẫu nhiên, xoay theo một góc ngẫu nhiên để cho ra hình ảnh thẻ căn cước đã được xoay.

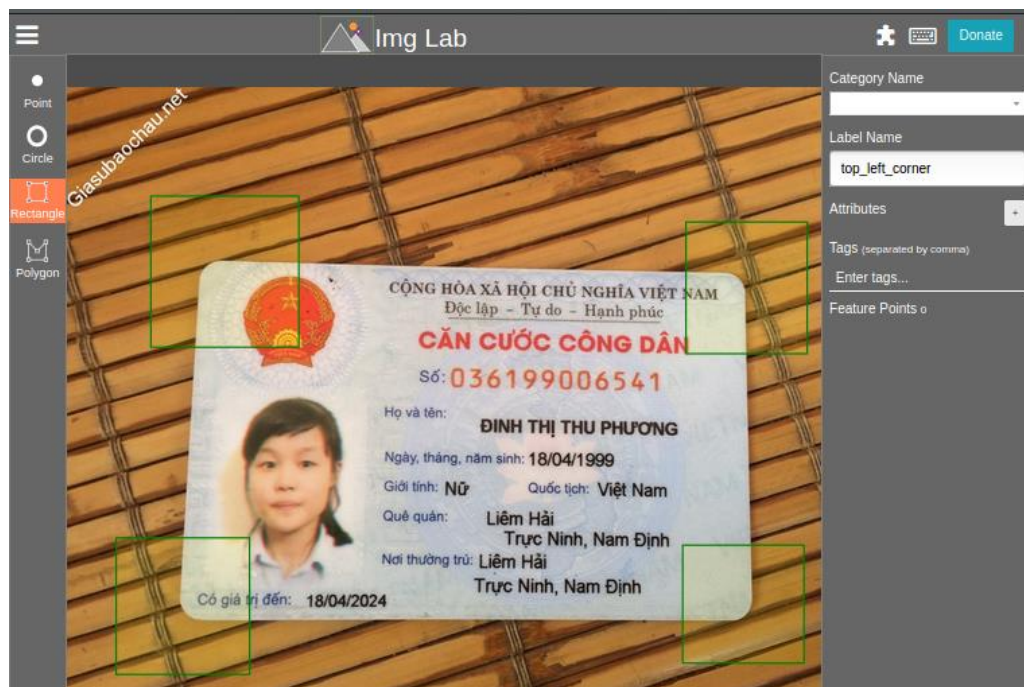


*Ảnh 57: Kết quả của quá trình tạo ảnh*

#### **4.2.1.2. Gán nhãn cho ảnh**

Sử dụng công cụ labelImg

LabelImg là một công cụ đồ họa mã nguồn mở được tạo ra phục vụ cho các bài toán nhận diện đối tượng, vật thể. Công cụ này có giao diện trực quan, người dùng chọn vùng có đối tượng cần nhận diện, đặt tên. labelImg sẽ tạo ra một tập tin có đuôi xml lưu lại thông tin hình ảnh, cũng như tọa độ của hình chữ nhật bao quanh đối tượng. Tập tin xml có định dạng giống với định dạng PascalVOC, một bộ dữ liệu nổi tiếng được sử dụng trong các bài toán trí tuệ nhân tạo liên quan đến hình ảnh. labelImg hỗ trợ nhiều nền tảng khác nhau như Windows, Mac, Web



Ảnh 58: Giao diện của công cụ labelImg trên nền tảng web

Tập tin lưu theo định dạng PascalVOC, lưu lại thông tin hình ảnh và tọa độ bốn hình chữ nhật chứa bốn góc phía trong. Nhóm đặt tên bốn góc cần nhận diện là: `top_left_corner` (góc trái trên), `top_right_corner` (góc phải trên), `bottom_left_corner` (góc trái dưới), `bottom_right_corner` (góc phải dưới).

Sau hai bước tạo dữ liệu bằng dữ liệu thật cũng như dữ liệu tạo bằng xử lý ảnh. Chúng ta có một tập dữ liệu hơn 1000 ảnh, với tọa độ của các đối tượng lưu lại dưới dạng xml

Đối với bước trích xuất các trường thông tin, cách tạo dữ liệu được thực hiện tương tự bằng labelIMG. Với tên các đối tượng được đặt như sau: `number` (số định danh), `name` (họ tên), `dob` (ngày sinh), `genre` (giới tính), `hometown` (quê quán), `address` (nơi thường trú)

#### 4.2.1.3. Tạo tập dữ liệu huấn luyện cho mô hình

Tensorflow huấn luyện mô hình với dữ liệu được lưu ở dạng tfrecord. Một định dạng do các nhà phát triển thư viện này quy định. Vì vậy để huấn luyện, dữ liệu cần được chuyển từ pascalVOC qua tfrecord.

raccoon\_dataset là một mã nguồn mở. Được xây dựng để tạo ra các bộ dữ liệu theo định dạng của tensorflow từ những bộ dữ liệu nổi tiếng khác như pascalVOC, fsns, yolo,...

Đầu tiên, từ file xml, thông tin của các đối tượng trong hình ảnh chuyển thành tập tin định dạng csv.

filename	width	height	class	xmin	ymin	xmax	ymax
1170.jpg	360	480	top_right_comer	295	127	338	172
1170.jpg	360	480	botom_right_comer	290	334	339	367
1171.jpeg	259	194	top_left_comer	3	18	34	44
1171.jpeg	259	194	botom_left_comer	1	165	18	191
1169.jpg	432	768	top_left_comer	303	140	365	210
1169.jpg	432	768	top_right_comer	292	494	361	560
1169.jpg	432	768	botom_left_comer	74	141	143	189
1169.jpg	432	768	botom_right_comer	74	495	150	571
1168.jpg	1024	768	top_left_comer	570	577	662	644
1168.jpg	1024	768	top_right_comer	593	102	655	184
1168.jpg	1024	768	botom_left_comer	855	578	955	645
1168.jpg	1024	768	botom_right_comer	888	104	968	182
1166.jpg	576	768	top_right_comer	514	157	576	204
1166.jpg	576	768	botom_left_comer	40	460	111	517
1166.jpg	576	768	botom_right_comer	537	460	576	502

Ảnh 59: Tập tin csv lưu tọa độ các hình chữ nhật chứa đối tượng

Từ tập tin csv, racoon\_dataset cung cấp một công cụ để chuyển dữ liệu thành một tập tin có phần mở rộng \*.record

#### 4.2.2. Huấn luyện mô hình

Do việc huấn luyện mô hình đòi hỏi cấu hình phần cứng mà các máy tính thông thường không đáp ứng được. Nhóm quyết định huấn luyện mô hình trên Google Colab (Colaboratory), một dịch vụ đám mây dùng để chạy các mã nguồn Python với môi trường được cài đặt sẵn, cũng như phần cứng cung cấp miễn phí.

Nhóm sử dụng các model Tensorflow cung cấp để huấn luyện. Ở cả hai tác vụ nhận diện 4 góc và nhận diện thông tin trên thẻ căn cước thì nhóm đều sử dụng mô hình SSD với bộ trích xuất ResnetFpn.

Quá trình huấn luyện mô hình kết thúc khi giá trị hàm mất mát đạt giá trị đủ nhỏ hoặc số bước được cài đặt trước đã thực thi hết.

```
+ Code + Text
I0718 10:47:30.273996 139637499393920 learning.py:507] global step 3: loss = 6.4083 (1.296 sec/step)
INFO:tensorflow:global step 3: loss = 10.7482 (1.407 sec/step)
... I0718 10:47:31.683276 139637499393920 learning.py:507] global step 3: loss = 10.7482 (1.407 sec/step)
INFO:tensorflow:global step 3: loss = 9.3692 (1.329 sec/step)
I0718 10:47:33.014489 139637499393920 learning.py:507] global step 3: loss = 9.3692 (1.329 sec/step)
INFO:tensorflow:global step 3: loss = 12.9787 (1.392 sec/step)
I0718 10:47:34.408544 139637499393920 learning.py:507] global step 3: loss = 12.9787 (1.392 sec/step)
INFO:tensorflow:global step 4: loss = 6.7179 (1.330 sec/step)
I0718 10:47:35.740364 139637499393920 learning.py:507] global step 4: loss = 6.7179 (1.330 sec/step)
INFO:tensorflow:global step 4: loss = 116.8978 (1.347 sec/step)
I0718 10:47:37.089428 139637499393920 learning.py:507] global step 4: loss = 116.8978 (1.347 sec/step)
INFO:tensorflow:global step 4: loss = 44.1917 (1.330 sec/step)
I0718 10:47:38.421473 139637499393920 learning.py:507] global step 4: loss = 44.1917 (1.330 sec/step)
INFO:tensorflow:global step 4: loss = 87.4271 (1.308 sec/step)
I0718 10:47:39.731770 139637499393920 learning.py:507] global step 4: loss = 87.4271 (1.308 sec/step)
INFO:tensorflow:global step 4: loss = 97.2008 (1.312 sec/step)
I0718 10:47:41.045212 139637499393920 learning.py:507] global step 4: loss = 97.2008 (1.312 sec/step)
INFO:tensorflow:global step 4: loss = 29.2273 (1.332 sec/step)
I0718 10:47:42.383080 139637499393920 learning.py:507] global step 4: loss = 29.2273 (1.332 sec/step)
INFO:tensorflow:global step 4: loss = 58.5352 (1.271 sec/step)
I0718 10:47:43.658809 139637499393920 learning.py:507] global step 4: loss = 58.5352 (1.271 sec/step)
INFO:tensorflow:global step 4: loss = 55.5822 (1.272 sec/step)
I0718 10:47:44.932532 139637499393920 learning.py:507] global step 4: loss = 55.5822 (1.272 sec/step)
INFO:tensorflow:global step 5: loss = 76.7902 (1.263 sec/step)
I0718 10:47:46.199105 139637499393920 learning.py:507] global step 5: loss = 76.7902 (1.263 sec/step)
```

Ảnh 60 Ảnh quá trình huấn luyện

Sau khi quá trình huấn luyện kết thúc, các trọng số được lưu thành tập tin có phần mở rộng \*.ckpt, từ tập tin này nhóm sẽ xuất ra một mô hình dùng để dự đoán vị trí của các đối tượng như 4 góc hay các đối tượng là thông tin trên thẻ căn cước.

### ➤ Khó khăn trong quá trình huấn luyện

Đầu tiên nhóm huấn luyện trên một mô hình nguyên thủy(scratch model) với các trọng số được khởi tạo ngẫu nhiên. Kết quả như hình trên có thể thấy được là giá trị của hàm mất mát rất lớn khiến việc huấn luyện mất rất nhiều thời gian để có thể đạt được một mô hình dự đoán tốt. Thư viện Tensorflow lại có hỗ trợ một số mô hình đã huấn luyện với bộ dữ liệu COCO2017 và nhóm quyết định sử dụng mô hình này để học chuyển(transfer learning). Lợi thế của phương pháp học chuyển này là từ mô hình đã huấn luyện với bộ dữ liệu trước ta có thể huấn luyện tiếp để mô hình học thêm các đối tượng mới mà chúng ta muốn nó phát hiện. Điều này giúp cho tốc độ huấn luyện nhanh hơn, kèm với đó là ta cần ít dữ liệu huấn luyện hơn.



```
+ Code + Text
10/18 05:25:25.41/880 140215773075328 learning.py:507] global step 0: loss = 3.5652 (0.549 sec/step)
2020-07-18 05:25:25.483620: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 573453216 exceeds 10% of system memory.
INFO:tensorflow:global step 1: loss = 3.0871 (0.785 sec/step)
...
I0718 05:25:26.283204 140215773075328 learning.py:507] global step 1: loss = 3.0871 (0.785 sec/step)
INFO:tensorflow:global step 1: loss = 2.0073 (0.612 sec/step)
I0718 05:25:26.897239 140215773075328 learning.py:507] global step 1: loss = 2.0073 (0.612 sec/step)
INFO:tensorflow:global step 1: loss = 1.7626 (0.527 sec/step)
I0718 05:25:27.426051 140215773075328 learning.py:507] global step 1: loss = 1.7626 (0.527 sec/step)
INFO:tensorflow:global step 1: loss = 2.7269 (0.668 sec/step)
I0718 05:25:28.095387 140215773075328 learning.py:507] global step 1: loss = 2.7269 (0.668 sec/step)
INFO:tensorflow:global step 1: loss = 1.8108 (0.576 sec/step)
I0718 05:25:28.673308 140215773075328 learning.py:507] global step 1: loss = 1.8108 (0.576 sec/step)
INFO:tensorflow:global step 1: loss = 2.8377 (0.582 sec/step)
I0718 05:25:29.256878 140215773075328 learning.py:507] global step 1: loss = 2.8377 (0.582 sec/step)
INFO:tensorflow:global step 1: loss = 2.6965 (0.670 sec/step)
I0718 05:25:29.929475 140215773075328 learning.py:507] global step 1: loss = 2.6965 (0.670 sec/step)
INFO:tensorflow:global step 1: loss = 2.3249 (0.544 sec/step)
I0718 05:25:30.474703 140215773075328 learning.py:507] global step 1: loss = 2.3249 (0.544 sec/step)
INFO:tensorflow:global step 1: loss = 2.1737 (0.523 sec/step)
I0718 05:25:30.999356 140215773075328 learning.py:507] global step 1: loss = 2.1737 (0.523 sec/step)
INFO:tensorflow:global step 2: loss = 2.2418 (0.578 sec/step)
```

Ảnh 61 Ảnh quá trình huấn luyện với mô hình đã được huấn luyện với bộ dữ liệu COCO2017

Như hình trên thì ta cũng thấy được là ngay từ đầu thì giá trị của hàm mất mát đã nhỏ ( chỉ trong khoảng 2-3 trong khi huấn luyện từ đầu thì giá trị lên đến trên 100).

### 4.3. Cài đặt thư viện pytesseract và ngôn ngữ việt cho pytesseract để đọc thông tin

Mô hình tesseract được hỗ trợ thành một thư viện của python có tên pytesseract và dễ dàng cài đặt

Pytesseract hỗ trợ nhiều ngôn ngữ khác nhau, người dùng cần cài đặt ngôn ngữ tương ứng để sử dụng. Để cài đặt người dùng tải bộ dữ liệu ngôn ngữ tương ứng từ trang web của tesseract, bộ dữ liệu có phần mở rộng \*.traineddata, chứa thông tin về ngôn ngữ cũng như các trọng số đã được huấn luyện, lưu vào thư mục cài đặt của tesseract

Thư viện pytesseract cung cấp sẵn hàm truyền vào tham số là đường dẫn của hình ảnh và ngôn ngữ, cho ra kết quả đầu ra là các ký tự trên ảnh, sắp xếp theo thứ tự từ trái qua phải, từ trên xuống dưới với ngôn ngữ là tiếng Việt

Tesseract cho kết quả có độ chính xác cao với hình ảnh đầu vào là chữ đen trên nền trắng, vì vậy hình ảnh trước khi đưa vào sẽ được tiền xử lý bằng các phương pháp nhị phân hóa, lọc nhiễu để cho kết quả tốt hơn

## 4.4. Cài đặt sửa chính tả cho tên, địa chỉ.

### 4.4.1. Sửa chính tả cho tên

Như ở chương 3 đã đề cập tới thì nhóm sử dụng xác suất để tìm từ phù hợp để chỉnh sửa.

$P(C|W) = P(W|C) \times P(C)$ ,  $P(C)$  được tính bằng cách lấy số lần xuất hiện của C trong từ điển chia cho tổng số từ trong từ điển vậy  $P(W|C)$  là độ khớp của W so với C được tính như thế nào.

Đầu tiên nhóm sử dụng một thư viện mang tên Fuzzywuzzy để tính độ khớp này. Thư viện này sử dụng Levenshtein Distance để tính toán độ lệch giữa hai chuỗi ký tự, độ lệch càng cao thì độ khớp càng thấp. Phương pháp tính khoảng cách giữa hai chuỗi A và B dựa vào số phép biến đổi 1 ký tự để đưa A về B (ví dụ 2 chuỗi là “Nguyễn” và “Nguye” sẽ có khoảng cách là 2 vì “Nguye” cần thay e thành ẽ và thêm 1 ký tự n để trở thành “Nguyễn”). Đây là phương pháp tính độ lệch giữa 2 chuỗi trong tiếng Anh khá chính xác vì tiếng Anh không có dấu, độ khác biệt giữa 2 ký tự bất kì là như nhau (độ lệch giữa a và b, a và c đều là 1) trong khi tiếng Việt thì không như vậy (ví dụ độ khác biệt giữa 2 ký tự a và á rõ ràng thấp hơn độ khác biệt giữa 2 ký tự a và b). Điều này khiến cho từ “TRÀNG” với 2 từ có thể thay thế là “TRANG” và “TRUNG”. Nếu dùng Levenshtein Distance thì  $P(W|C)$  của hai trường hợp này là như nhau, trong khi rõ ràng ta thấy  $P(W|C)$  của “TRANG” rõ ràng phải cao hơn trong trường hợp này. Vì vậy nhóm quyết định viết lại một phương thức tính toán khoảng cách mới thay cho Levenshtein Distance

bằng cách tạo ra một bảng mã như sau:

'A': 0,	'Á': 1,	'À': 2,	'Ạ': 3,	'Ã': 4,	'Ả': 5,
'Ă': 11,	'Ằ': 12,	'Ẳ': 13,	'Ẵ': 14,	'Ẻ': 15,	'Ẻ': 16,
'Â': 22,	'Ấ': 23,	'Ằ': 24,	'Ậ': 25,	'Ằ': 26,	'Ả': 27,
'B': 55,	'C': 83,	'D': 111,	'Đ': 112,		
'E': 140,	'É': 141,	'È': 142,	'Ẻ': 143,	'Ễ': 144,	'Ẻ': 145,
'Ê': 151,	'Ế': 152,	'Ề': 153,	'Ẻ': 154,	'Ễ': 155,	'Ẻ': 156,
'F': 184,	'G': 212,	'H': 240,			
'I': 268,	'Í': 269,	'Ì': 270,	'Ị': 271,	'Ĩ': 272,	'Ỉ': 273,
'J': 301,	'K': 329,	'L': 357,	'M': 385,	'N': 413,	
'O': 441,	'Ó': 442,	'Ò': 443,	'Ọ': 444,	'Õ': 445,	'Ỏ': 446,
'Ô': 452,	'Ố': 453,	'Ỗ': 454,	'Ộ': 455,	'Ỗ': 456,	'Ổ': 457,
'Ơ': 463,	'Ớ': 464,	'Ờ': 465,	'Ợ': 466,	'Ỗ': 467,	'Ở': 468,
'P': 496,	'Q': 524,	'R': 552,	'S': 580,	'T': 608,	
'U': 636,	'Ú': 637,	'Ù': 638,	'Ụ': 639,	'Ũ': 640,	'Ủ': 641,
'Ư': 647,	'Ứ': 648,	'Ừ': 649,	'Ự': 650,	'Ữ': 651,	'Ử': 652,
'V': 680,	'X': 708,				
'Y': 736,	'Ý': 737,	'Ỡ': 738,	'Ỡ': 739,	'Ỡ': 740,	'Ỡ': 741

Ảnh 62 Bảng mã cho các kí tự tiếng Việt

Cách tính khoảng cách 2 ký tự sẽ là lấy giá trị tuyệt đối của hiệu 2 kí tự. Sau đó nếu giá trị này lớn hơn 27 ( từ A đến Æ hoặc giữa 2 kí tự khác nhau bất kì mà không phải là đối dấu như B và C) thì khoảng cách sẽ là  $\max = 741$ , nếu giá trị này lớn hơn 5 (như A và Æ hoặc O và Ô) thì khoảng cách sẽ là 300 và nếu giá trị này bé hơn 5 thì khoảng cách chính là giá trị này. Với cách tính này rõ ràng đã giúp cho “TRANG” gần với “TRÀNG” hơn là “TRUNG” với “TRÀNG”. Sau khi tính toán được  $P(W|C)$  từ phương pháp trên cùng với  $P(C)$  tính được, ta tính được  $P(C|W)$  của tất cả các từ có thể thay thế cho W sau đó chọn ra từ C có giá trị  $P(C|W)$  lớn nhất.

#### 4.4.2. Sửa chính tả cho địa chỉ

Nhóm cài đặt theo đúng thiết kế đã trình bày ở chương 3 và sử dụng Levenshtein Distance để tính độ lệch vì độ lệch giữa các tên các đơn vị hành chính là lớn và không cần thiết sử dụng cách tính khoảng cách như ở tên.

#### **4.5. Tạo một API nhận vào một ảnh và trả ra các thông tin trên thẻ căn cước**

Sau khi huấn luyện các mô hình nhóm sẽ sử dụng thư viện Flask để tạo một API như sau. Đầu vào sẽ nhận vào một ảnh chứa thẻ căn cước và đưa ảnh đó qua mô hình nhận diện 4 góc. Sử dụng kết quả nhận được để cắt thẻ căn cước ra khỏi nền và đưa vào mô hình nhận diện các thông tin. Sau khi có được vị trí của các thông tin của thẻ căn cước thì nhóm tiến hành cắt các phần ảnh chứa thông tin của thẻ ra và đưa lần lượt vào mô hình đọc ký tự từ ảnh. Sau khi qua bước đọc thông tin từ các ảnh thì nhóm nhận về các chuỗi thông tin trên thẻ căn cước và đưa qua mô-đun sửa chính tả để sửa lại cái chuỗi này. Cuối cùng là trả về cho người dùng những thông tin trên.

# Chương 5: Tổng kết

## 5.1 Kiến thức thu được

Trước khi thực hiện khoá luận này thì hai thành viên của nhóm đều là những sinh viên của chuyên ngành kỹ thuật phần mềm và chưa từng học những kiến thức liên quan đến trí thông minh nhân tạo nên hầu hết các kiến thức được trình bày trong luận văn đều là kiến thức mới đối với nhóm. Từ những lý thuyết về các mạng nơron, kiến thức về xử lý ảnh hay việc sử dụng thư viện Tensorflow trong máy học đều là kiến thức mới.

## 5.2 Sản phẩm thu được

Sản phẩm thu được của khoá luận là tài liệu nghiên cứu về giải pháp cho bài toán đọc thông tin trên thẻ căn cước công dân Việt Nam cùng một ứng dụng áp dụng giải pháp trên dưới dạng một API. Các ứng dụng khác có thể tích hợp giải pháp này để có thể tạo ra những sản phẩm có ý nghĩa với đời sống.

## 5.3 Đánh giá độ chính xác của sản phẩm

Độ chính xác của sản phẩm phụ thuộc rất nhiều vào chất lượng hình ảnh, cải thiện được so với các giải pháp đã có trước đó như BeeOCR. Tuy nhiên đối với những ảnh chất lượng thấp hoặc bị ảnh bị chói sáng khiến mất thông tin thì ảnh chưa xử lý tốt được như giải pháp của FPT Vision

Kết quả ban đầu của mô hình sau khi áp dụng để kiểm tra 100 ảnh như sau, tỉ lệ chính xác được tính bằng tỉ lệ từ trùng khớp giữa kết quả nhận diện và kết quả thực tế



Tỉ lệ chính xác	Số lượng ảnh
100%	21
90% - 100%	42
80% - 90%	22
<80%	15

*Bảng thống kê kết quả kiểm thử*

## Demo nhận diện thẻ căn cước



CHỌN ẢNH

### Kết quả

Số id: 036300010238  
Họ tên: VŨ HOÀI THU  
Giới tính: Nữ  
Ngày sinh: 09/10/2000  
Quê quán: Hải Bắc , Hải Hậu , Nam Định  
Địa chỉ: Hải Bắc , Hải Hậu , Nam Định

*Ảnh 63 Nhận diện chính xác 100%*

### Demo nhận diện thẻ căn cước



CHỌN ẢNH

#### Kết quả

Số id: 6134 446

Họ tên: NGUYỄN THỊ LINH

Giới tính: Nữ

Ngày sinh: 31/05/2000

Quê quán: Cao Thắng, Thanh Miện, Hải Dương

Địa chỉ: Cao Thắng, Thanh Miện, Hải Dương

*Ảnh 64 Nhận diện chính xác trên 90% sai một ít ở trường số căn cước*

## 5.4 Phương hướng phát triển và nghiên cứu trong tương lai

Sản phẩm khoá luận thu được có sử dụng mô hình đọc ký tự từ pytesseract và dựa trên dữ liệu huấn luyện của chính thư viện này khiến cho nhóm chưa kiểm soát được độ chính xác của mô hình. Trong tương lai nhóm sẽ nghiên cứu thêm và tự huấn luyện một mô



hình đọc ký tự khác để có thể cải thiện thêm được độ chính xác cho bài toán này. Khoa luận cũng có thể phát triển thêm một mô hình xử lý ngôn ngữ tự nhiên để có xử lý việc sửa chính tả tốt hơn cùng với đó là đọc cả thông tin mặt sau của thẻ căn cước. Ngoài ra nhóm cũng hướng đến việc nghiên cứu về nhận diện khuôn mặt để có thể tích hợp đọc thông tin thẻ căn cước cũng như xác định được thẻ căn cước đó có chính chủ không.

# Phụ lục

## Giới thiệu Colab

Hiện nay, các thuật toán machine learning, deep learning thường yêu cầu phần cứng có tốc độ xử lý cao (thường là GPU hoặc TPU). Các máy tính thông thường thường không được trang bị sẵn những phần cứng đáp ứng được nên gây ra những khó khăn nhất định trong việc xây dựng, phát triển các ứng dụng liên quan đến trí tuệ nhân tạo

Google colab (Colaboratory) là một dịch vụ miễn phí được cung cấp bởi Google, dùng để chạy mã nguồn python trên trình duyệt với môi trường được cài đặt sẵn.

Colaboratory là dịch vụ đám mây, dùng trực tiếp trên trình duyệt có kết nối internet.

Google colab được phát triển dựa trên Jupyter notebook, một công cụ được sử dụng để viết mã nguồn python nên cách sử dụng tương tự Jupyter notebook.

Google colab cung cấp GPU, TPU miễn phí cho người phát triển, hỗ trợ RAM 12GB.

Cấu hình mạnh hơn phần lớn các máy tính thông thường. Môi trường để viết mã nguồn và thực thi các lệnh được viết bằng python được cài đặt sẵn. Đồng thời Colaboratory cũng tích hợp các thư viện phổ biến thường được dùng trong nghiên cứu như Tensorflow, keras, PyTorch hay OpenCV

Tuy nhiên, Google Colab tồn tại một số hạn chế, người sử dụng cần lưu ý để sử dụng có hiệu quả. Do nguồn tài nguyên Google cung cấp là miễn phí và có giới hạn nên tại một số thời điểm, GPU, TPU đã được sử dụng hết và người dùng phải chờ đợi. Một hạn chế khác là dịch vụ đám mây này không hoạt động liên tục mà có giới hạn thời gian sử dụng, khi dịch vụ ngừng hoạt động. Các dữ liệu được lưu trữ cục bộ sẽ bị mất, do đó người dùng cần thường xuyên lưu lại các giá trị quan trọng vào bộ nhớ của Google drive - một dịch vụ lưu trữ trực tuyến được liên kết với Colaboratory. Ngoài ra, Google Colab hiện tại chỉ hỗ trợ hai phiên bản Python là 2.7 và 3.6

# Tài liệu tham khảo

- [1] T. N. T. Thanh, “A Method for Segmentation of Vietnamese Identification Card Text Fields,” 2019.
- [2] R. Girshick, J. Donahue, T. Darrell và J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” 2014.
- [3] R. Girshick, “Fast R-CNN,” 2015.
- [4] S. Ren, K. He, R. Girshick và J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” 2016.
- [5] J. Redmon, S. Divvala, R. Girshick và A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016.
- [6] L. Wei, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu và A. C. Berg, “SSD: Single Shot MultiBox Detector,” 2016.
- [7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto và H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.
- [8] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan và S. Belongie, “Feature Pyramid Networks for Object Detection,” 2017.