TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education

HCMUTE

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

BÁO CÁO CUỐI KỲ
MÔN HỌC TRÍ TUỆ NHÂN TẠO

LÝ THUYẾT VÀ ỨNG DỤNG CONVOLUTIONAL NETWORKS VÀO NHẬN DIỆN KHUÔN MẶT CÓ ĐEO KHẨU TRANG

GVDH: TS. NGUYỄN TRƯỜNG THỊNH

SHTH: PHẠM CÔNG THÀNH – MSSV: 18146213

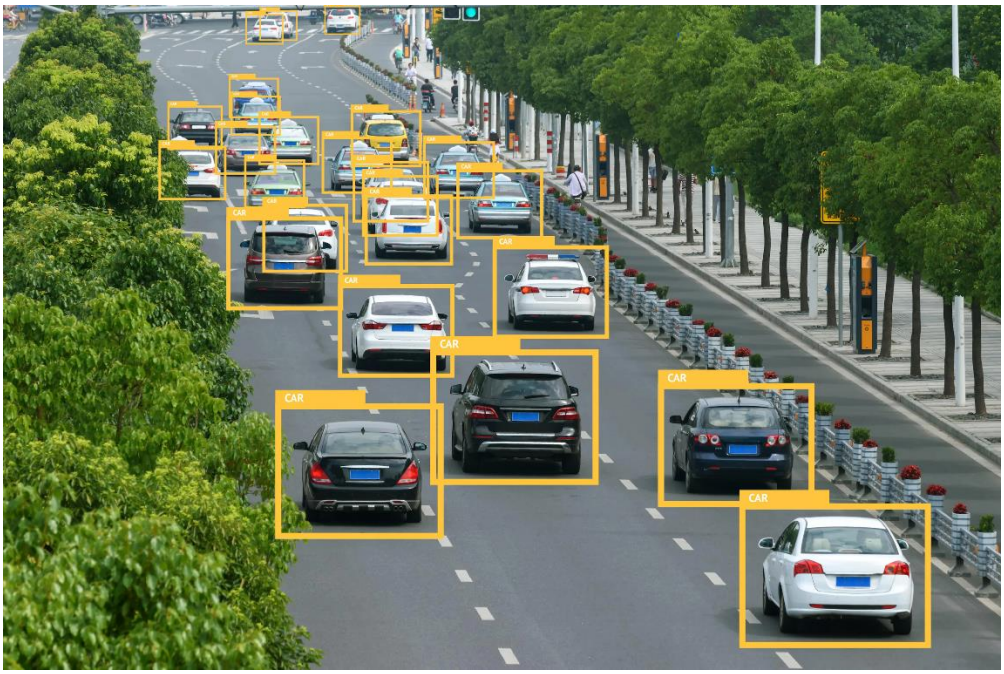
Tp. Hồ Chí Minh, tháng 06 năm 2020

MỤC LỤC:

CHƯƠNG 1: TỔNG QUAN

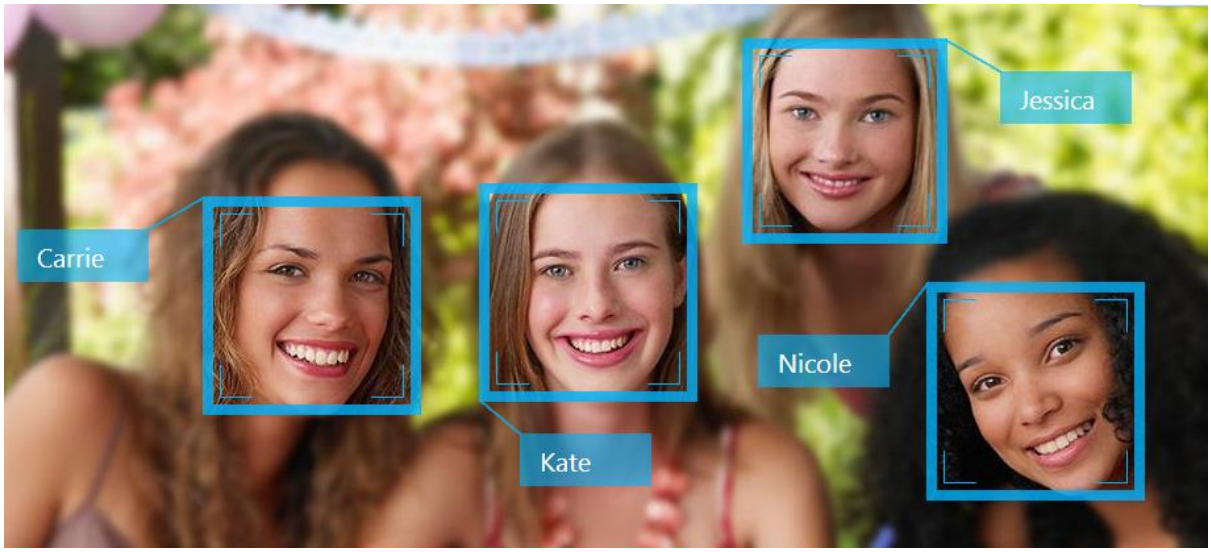
1.1. Đặt vấn đề.

Bài toán nhận diện khuôn mặt có thể coi là một trường hợp cụ thể của bài toán nhận diện vật thể (object detection). Trong bài toán nhận diện vật thể, nhiệm vụ chính là tìm ra vị trí và kích thước của vật thể theo yêu cầu. Ví dụ như bài toán xác định nhưng chiếc xe ô tô có trong hình, chúng ta sẽ có kết quả sau:



Hình 1.1. Nhận diện xe ô tô có trong hình (car detection).

Hay bài toán nhận diện khuôn mặt, mục tiêu chính là xác định có sự xuất hiện của khuôn mặt con người hay không, vị trí xuất hiện của khuôn mặt và cả kích thước khuôn mặt nếu cần thiết:



Hình 1.2. Nhận diện khuôn mặt người (face detection).

Còn rất nhiều những ứng dụng khác của bài toán nhận diện vật thể, nhưng mục tiêu của bài báo cáo này là nhận diện khuôn mặt. Nhận diện khuôn mặt xuất từ nhu cầu nhận diện con người, phân biệt giữa người với người nhằm mục đích quản lý, bảo vệ an ninh xã hội ,… như xác định tội phạm.

Nhu cầu nhận diện con người, phân biệt từng chủ thể con người đã xuất hiện từ rất lâu. Vào thế kỷ thứ II trước công nguyên, người Trung Quốc đã sử dụng mực để lấy và lưu lại dấu vân tay nhằm mục đích xác định từng người, chúng ta có nghe rất quen với từ điểm chỉ hay lăn tay, đó chính là việc lấy dấu vân tay.

Hình 1.3. Điểm chỉ lấy dấu vân tay ở Trung Quốc.

Cột mốc quan trọng tiếp theo đó là từ năm 1880 đến năm 1914, Alphonse Bertillon là một người tiên phong trong lĩnh vực khoa học pháp y, ông đã cho ra những sáng kiến về việc xác định danh tính một người bằng 11 chỉ số được đo bằng các đặc tính trên con người như màu mắt, màu da,… Vào năm 1988, ông còn phát minh “mugshot” toàn mặt, hiểu trong tiếng việt là hình cảnh sát hay hình khi bắt. Ở hiện tại, chúng ta có thể rất quen với việc các phạm nhân sau khi bị bắt sẽ phải chụp 2 bức ảnh để phía cảnh sát lưu lại hồ sơ, đó chính là mugshot, hình ảnh dưới đây sẽ là một ví dụ minh họa dễ hiểu cho bạn:

Hình 1.4. Mugshot toàn mặt của chính Alphonse Bertillon.

Tuy nhiên có một số trở ngại, có thể bạn biết về khái niệm song sinh cùng trứng, khi sinh ra, hai đứa trẻ sẽ mang vẻ bề ngoài cực kỳ giống nhau, sự việc này sẽ gây trở ngại lớn cho bài toán xác định danh tính. Cùng với đó là việc gia tăng dân số, số lượng người quá lớn nên việc xác định danh tính một người bằng sinh trắc học là một thử thách cực kỳ lớn, bởi việc đó yêu cầu sự chấp thuật của người khác để có thể lấy mẫu máu, chụp ảnh mugshot, đánh giá từng đặc tính của họ. Chính vì thế mà bài toán nhận diện khuôn mặt tự động sẽ là giải pháp cực kỳ hiệu quả, chúng ta chỉ cần hình ảnh và sau đó sử dụng các thuật toán để xác định xem có khuôn mặt người xuất hiện hay không.

Hệ thống nhận diện khuôn mặt tự động đầu tiên được phát triển vào năm 1973, bởi Takeo Kanade. Mặc dù ý tưởng về hệ thống tự động nhận diện khuôn mặt này được đưa ra và nghiên cứu phát triển lần đầu tiên bởi Bledsoe, Helen Chan Wolf, Charles Bisson vào những năm 1960. Tuy nhiên, cỗ máy mà những nhà tiên phong này làm được coi là "máy chạy bằng cơm", bởi vì các tọa độ của các đặc tính trên khuôn mặt như tâm đồng tử của mắt đều phải do con người thiết lập trước khi cho máy nhận dạng. Ở thời điểm những năm 1960, một người có thể xác định vị trí đặc tính của khuôn mặt trên 40 hình ảnh trong một giờ, và khoảng 13 năm sau đó chúng ta đã có một bước tiến nữa khi chúng ta có thể làm ra một hệ thống tự động. Tuy nhiên, vẫn còn những khó khăn về cơ sở dữ liệu khuôn mặt người và các kỹ thuật máy tính khiến cho kết quả đạt được của hệ thống mà Takeo Kanade tạo ta không đạt được hiểu quả, không có sự chính xác như mong muốn. Do những vấn đề nan giải đó mà ý tưởng hệ thống nhận dạng khuôn mặt tự động đã bị bỏ ngỏ. Đến tận những năm 1990, bằng cách áp dụng phép biến đổi Karhunen-Loeve, Kirby và Sirovich đã nghiên cứu thành công và làm giảm số chiều biểu diễn cho bức ảnh khuôn mặt người. Và công trình nghiên cứu "eigenface" của Turk và Pentland đã làm sống lại chủ đề nhận diện khuôn mặt tự động. Kể từ thời điểm đó đến nay, nhân loại đã sáng tạo thêm rất nhiều kỹ thuật khác nhau nhằm đáp ứng nhu cầu nhận diện khuôn mặt trong từng trường hợp cụ thể.

1.2. Mục tiêu.

Bài toán nhận diện khuôn mặt là một bài toán cực kỳ khó khăn nếu chúng ta muốn kết quả đạt được nhanh và chính xác. Từ năm 1973 đến nay, đã có rất nhiều thuật toán, kỹ thuật khác được phát triển nhằm đáp ứng nhu cầu nhận diện khuôn mặt của con người.

Với tình hình dịch bệnh CoVID kéo dài, mọi người đều phải đeo khẩu trang nhằm bảo vệ sức khỏe cộng đồng cũng như bảo vệ bản thân. Khẩu trang là trang bị cá nhân được đeo trên khuôn mặt, chúng ta có thể thấy rõ trong hình ảnh sau:

Hình 1.5. Mặt người khi đeo khẩu trang.

Để xác định trong một nhóm người trong một phạm vi nhất định có bao nhiêu người không đeo khẩu trang là nhu cầu cấp thiết với tình hình dịch bệnh hiện tại. Chính vì thế, mục tiêu của đồ án môn học này là làm một hệ thống nhận diện khuôn mặt của một người có đeo khẩu trang hay không đeo khẩu trang rồi tiến hành thông báo và các biện pháp khuyến cáo.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về nhận diện khuôn mặt

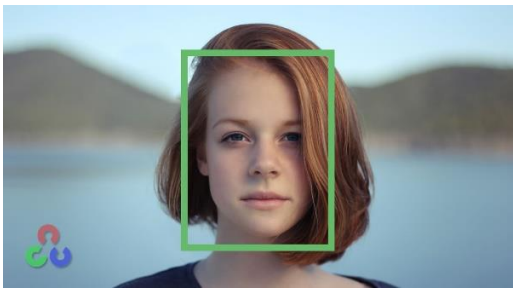
Hệ thống nhận diện khuôn mặt được ứng dụng rất nhiều trong hiện tại, tùy vào từng yêu cầu cụ thể của người sử dụng mà có thể kết quả đầu ra sẽ khác nhau. Tuy nhiên, khi nói về các hệ thông nhận diện khuôn mặt người, chúng ta thể phân nó thành hai loại với hai mục đích chính:

- Phát hiện khuôn mặt.
- Xác định hay nhận dạng khuôn mặt (nhằm xác định danh tính).

Trong cả hai trường hợp trên chúng ta đều cần có một cơ sở dữ liệu lớn về hình ảnh có khuôn mặt người làm dữ liệu đầu vào cho hệ thống học. Kèm theo đó là tập dữ liệu hình ảnh thể kiểm tra đánh giá kết quả của hệ thống sau khi học. Khi kết quả chấp nhận được thì chúng ta có thể sử dụng hệ thống đó.

Nói đến hệ thống phát hiện khuôn mặt, hệ thống chỉ cần so sánh các giá trị, thông tin từ hình ảnh cần kiểm tra với hình ảnh trong cơ sở dữ liệu mà máy đã học, quá trình này được thực hiện thông qua thuật toán, hay còn gọi là giải thuật.

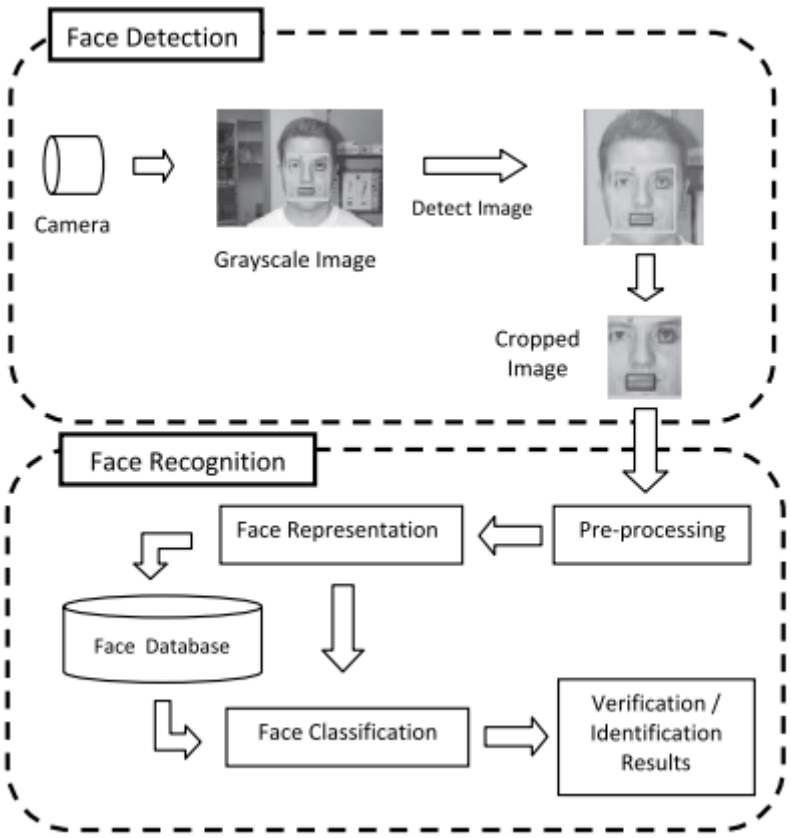
Còn ở mảng ứng dụng khác của hệ thống nhận diện khuôn mặt người, nhận dạng hay xác định khuôn mặt người, sẽ có phần phức tạp hơn. Bước đầu tiên hệ thống vẫn cần phải nhận diện xem có khuôn mặt người trong ảnh hay không. Sau đó hệ thống sẽ đối chiếu khuôn mặt đang kiểm tra so với các khuôn mặt có trong cơ sở dữ liệu, nếu đã có khuôn mặt đó trong cơ sở dữ liệu, hệ thống sẽ trả về kết quả có và ngược lại.



Hình 2.1. Phát hiện khuôn mặt người.

Hình 2.2. Xác định khuôn mặt người (xác định danh tính).

Hình ảnh sơ đồ khối dưới đây mô tả quá trình phát hiện và xác định khuôn mặt người của một hệ thống trong thực tế:



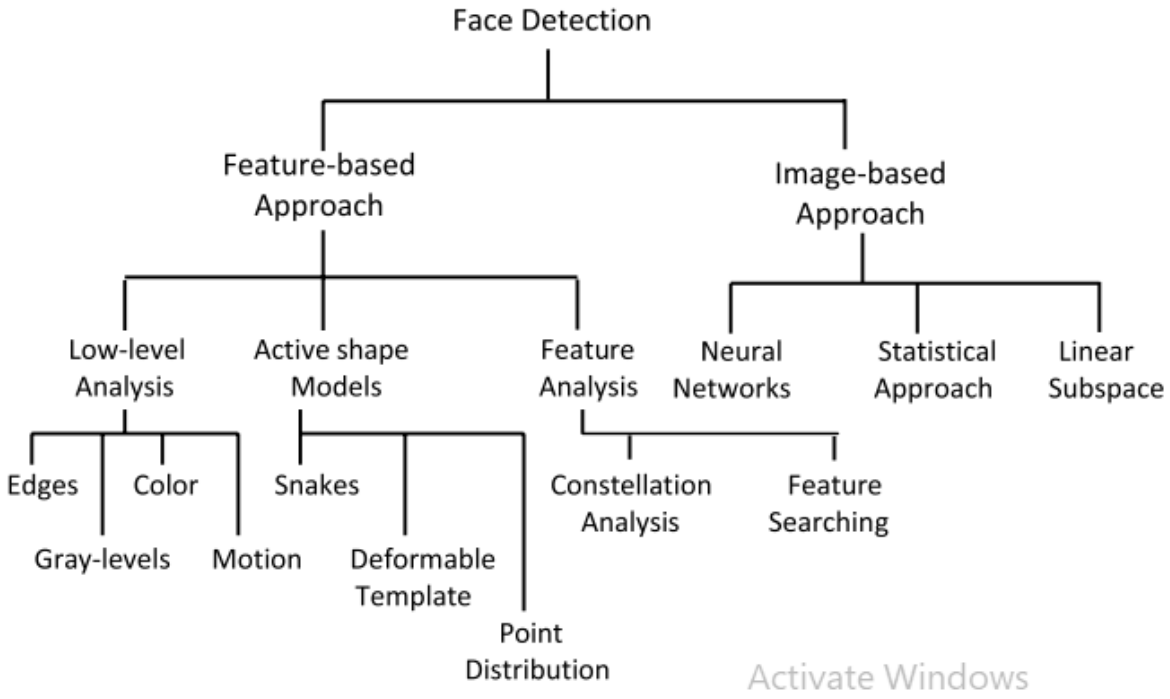
Hình 2.3. Sơ đồ khối về quá trình nhận dạng và xác định khuôn mặt

2.2. Sơ lược về các kỹ thuật nhận diện khuôn mặt

2.2.1. Face detection – Phát hiện khuôn mặt

Để miêu tả một khuôn mặt người chúng ta cần rất nhiều đặc tính của khuôn mặt đó, mà mỗi người lại có khuôn mặt nhau, điều đó khiến cho việc nhận diện khuôn mặt trở nên khó khăn. Bên cạnh đó, nếu dùng một hệ thống để nhận diện, bản thân hệ thống đó cũng có những giới hạn nhất định như giới hạn về phần cứng, giới hạn về phần mềm. Do đó việc phát triển một hệ thống nhận diện khuôn mặt đáp ứng kết quả tốt vẫn là một đề tài thu hút những nhân tài của nhân loại.

Sau vài thập niên nghiên cứu và phát triển, các chuyên gia tiên phong trong lĩnh vực nhận diện khuôn mặt đã đưa ra hơn 150 bài báo cáo với những hướng tiếp cận khác nhau. Sơ đồ dưới đây sẽ tổng hợp các hướng tiếp cận đó:



Hình 2.4. Các kỹ thuật phát hiện khuôn mặt.

2.2.2. Face recognition – Nhận dạng khuôn mặt

Hơn 20 năm qua, đề tài nhận dạng khuôn mặt phát triển song song với nhận diện khuôn mặt, đã có rất nhiều kỹ thuật được phát minh nhằm tối ưu kết quả đầu ra. Trong tất cả các thuật toán mặt đã có, chúng ta có thể phân thành 5 loại dựa trên cách mà các thuật toán nhận dạng khuôn mặt:

- Nhận dạng dựa trên xét tổng thể khuôn mặt (Appearance-feature based).
- Nhận diện khuôn mặt dựa vào các không gian con (Subspace based).
- Các kỹ thuật sử dụng neural networks.
- Nhận dạng dựa trên các mô hình (Model based).
- Một số thuật toán nhận dạng dựa vào các hệ số tương quan (correlation) và máy vecto hỗ trợ (support vector machines).

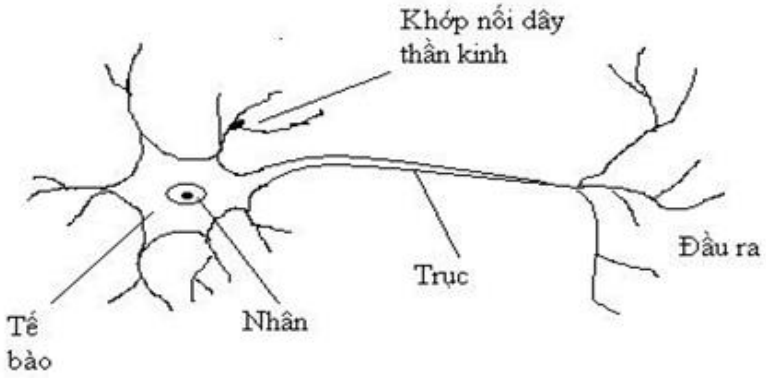
2.3. Cơ sở lý thuyết của về mạng nơ ron nhận tạo

Trong đồ án nhận dạng khuôn mặt người có đeo khẩu trang này, thuật toán được sử dụng là các mạng nơ ron, cụ thể hơn là mạng nơ ron tích chập (CNN – Convolutional Neural Networks). Mạng nơ ron tích chập là một thuật toán tiêu biểu trong khối kiến thức đồ sộ về Học sâu (Deep Learning). Những cơ sở lý thuyết sẽ được diễn giải chi tiết ở các mục dưới.

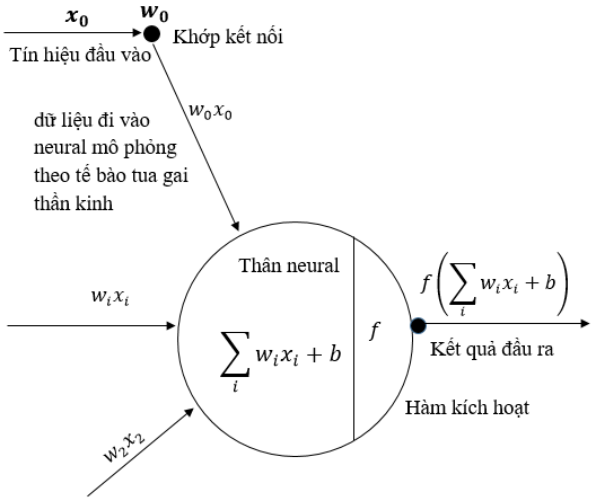
2.3.1. Giới thiệu về một nơ ron nhân tạo

Mục tiêu của một nơ ron nhân tạo khi được tạo ra là có thể bắt chước khả năng tự học và phân tích nhiều vấn đề khác nhau như một nơ ron thật trong bộ não con người.

Hai hình sau sẽ cho ta cái nhìn tổng quan về nơ ron nhân tạo và nơ ron thực được mô hình hóa:

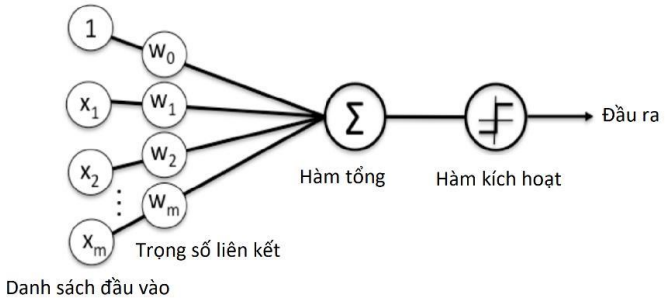


Hình 2.5. Mô hình của một nơ ron sinh học.



Hình 2.6. Mô hình cấu tạo của một nơ ron nhân tạo.

Chúng ta có thể biểu diễn nơ ron nhân tạo một cách đơn giản hơn:



Hình 2.7. Mô hình đơn giản của một nơ ron nhân tạo.

Hình 2.7. còn được biết đến là mô hình Perceptron, một mô hình nơ ron nhân tạo sơ khai, nền tảng cho các mạng nơ ron phức tạp sau này.

Tương tự như nơ ron thật, trong mô hình Perceptron, các xung thần kinh được truyền cho nơ ron bây giờ được mô phỏng bằng các giá trị đầu vào (danh sách đầu vào).

Các trọng số liên kết tượng trưng cho sự liên kết giữa các nơ ron. Khi các nơ ron thực truyền dữ liệu cho nhau, chúng sẽ truyền xung điện thông qua các trục. Đối với mô hình Perceptron, trọng số liên kết thể hiện cho độ mạnh yếu của giá trị mà nơ ron ở lớp trước đưa vào nơ ron ở lớp sau. Ở mô hình trên chúng ta có 2 lớp nơ ron, lớp thứ nhất gồm các nơ ron đầu vào và lớp thứ hai là nơ ron đầu ra.

Hàm kết hợp sẽ tính tổng các tích của trọng số với giá trị đầu vào của nơ ron lớp trước ($w_j.x_j$), trong đó w_j là các trọng số, còn x_j là giá trị đầu vào của các nơ ron.

Một điều đáng lưu ý, giá trị 1 ở nơ ron có liên kết với trọng số w_0, giá trị này được gọi là độ lệch (bias feature). Độ lệch được thêm vào trong phương trình để điều chỉnh đầu ra cùng với tổng trọng số các đầu vào cho nơ ron, giúp dịch chuyển giá trị của hàm kích hoạt sang trái hoặc phải, điều này sẽ giúp ích cho các mô hình huấn luyện của máy.

2.3.2. Các hàm kích hoạt trong mạng nơ ron nhân tạo

Bản chất của các hàm kích hoạt trong mạng nơ ron nhân tạo là các hàm phi tuyến. Các vấn đề mà chúng ta muốn máy giải quyết bằng mạng nơ ron rất phức tạp. Nếu chúng ta bỏ qua các hàm kích hoạt này thì các giá trị đầu vào đầu ra của nơ ron cũng chỉ nhân với các trọng số liên kết, điều này sẽ làm sai kết quả đầu ra cuối cùng của mạng nơ ron.

Có rất nhiều hàm kích hoạt được phát triển và càng ngày càng được tối ưu bởi những chuyên gia, một trong số đó là hàm tiêu biểu sigmoid. Trong các mục tiếp theo, chúng ta sẽ tìm hiểu về một số hàm kích hoạt tiêu biểu.

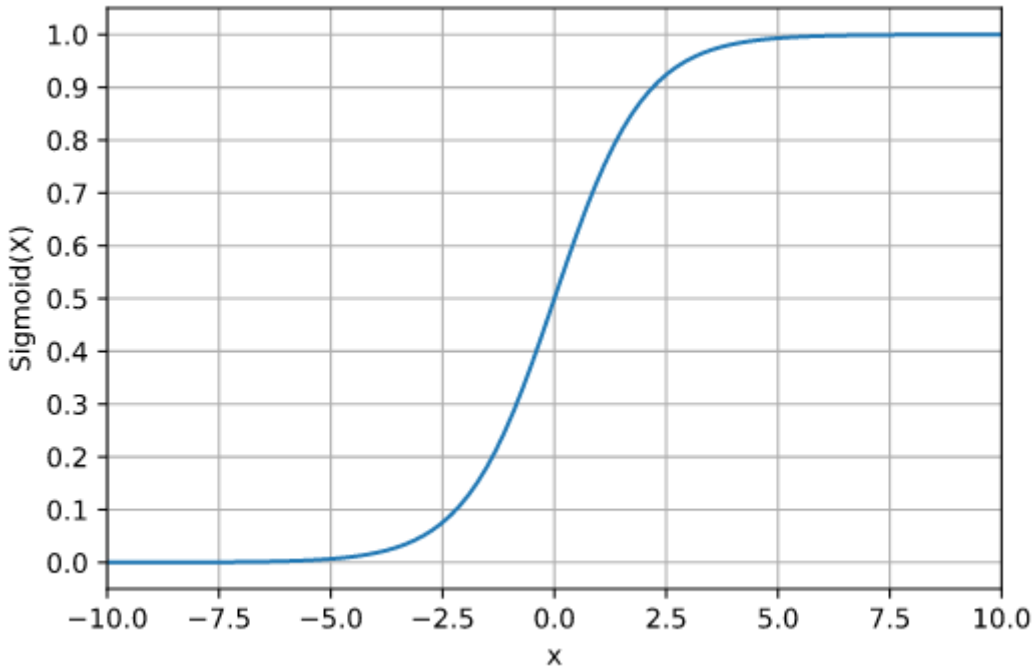
2.3.2.1. Hàm Sigmoid

Công thức toán và đạo hàm của hàm Sigmoid:

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

$$f'(x) = f(x)(1 - f(x))$$

Hàm Sigmoid được ứng dụng làm hàm kích hoạt (activate function) bởi vì giá trị trả về của nó nằm trong khoảng (0, 1) khi chúng ta cho một giá trị thực ở đầu vào (giá trị x, trên trục x). Hàm này được ứng dụng nhiều trong các bài toán dự đoán xác suất của kết quả đầu ra, như trong lĩnh vực toán học xác suất, các giá trị xác suất của một vấn đề chỉ nằm trong khoảng (0, 1). Đồ thị dưới đây biểu diễn hàm Sigmoid.



Hình 2.6. Đồ thị hàm Sigmoid.

Như ta đã thấy trên đồ thị, khi giá trị x (giá trị đầu vào) rất lớn và tiến đến dương vô cùng thì giá trị của sigmoid(x) sẽ xấp xỉ 1. Tương tự, khi giá trị của x rất bé tiến về âm vô cùng thì giá trị của sigmoid(x) sẽ xấp xỉ 0.

Khi dùng trong mạng nơ ron, giá trị sigmoid(x) ≥ 0 thì nơ ron sẽ mang trạng thái kích thích. Khi giá trị sigmoid(x) < 0 thì nơ ron sẽ mang trạng thái kiềm chế.

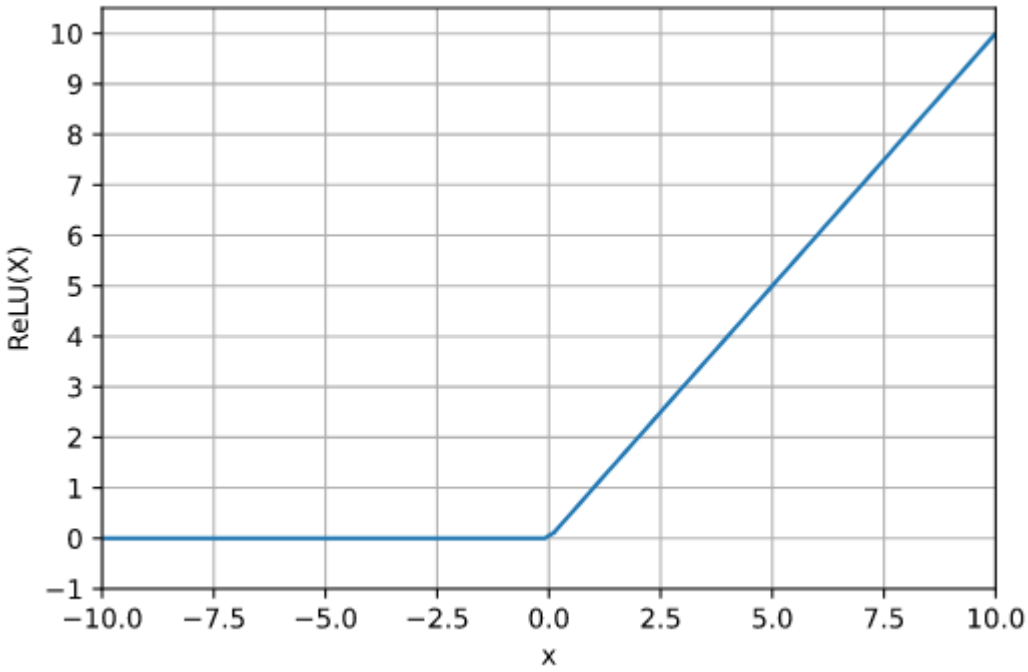
Tuy rằng hàm Sigmoid đã từng được sử dụng phổ biến, nhưng vì hai hạn chế lớn mà ngày nay sigmoid không còn được ứng dụng trong các mô hình học máy nữa. Hạn chế thứ nhất là hàm Sigmoid rất khó hội tụ. Nhược điểm thứ hai là hàm Sigmoid sẽ làm bão hòa mô hình học máy.

2.3.2.2. Hàm ReLU

Một hàm kích hoạt tiêu biểu khác đó chính là ReLU – Rectified linear unit. Hàm ReLU đang được sử dụng phổ biến trong thời gian gần đây. Bởi vì ReLU có một số ưu điểm so với các hàm tiền nhiệm. Ưu điểm đầu tiên mà bạn có thể thấy rõ là hàm ReLU rất dễ hiểu. Công thức của hàm ReLU như sau:

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

Và dưới đây là đồ thị của hàm ReLU:



Hình 2.7. Đồ thị của hàm ReLU.

Ưu điểm thứ hai của hàm ReLU là nó hội tụ nhanh hơn rất nhiều so với hàm Sigmoid. Và hàm ReLU được thực thi nhanh hơn do công thức rất đơn giản, do đó sẽ tốn ít tài nguyên hơn để thực thi.

Tuy nhiên, hàm kích hoạt ReLU vẫn còn một số nhược điểm như:

- Khi các giá trị $x \leq 0$ thì hàm ReLU sẽ cho ra giá trị là 0. Nếu các giá trị từ nơ ron thành 0 thì sẽ không có ý nghĩa với các bước linear activation tiếp theo.
- Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật.

2.3.3. Mạng nơ ron tích chập

Convolutional Neural Networks (CNNs) là một trong những mô hình deep learning tiên tiến đang được sử dụng phổ biến trong thời gian gần đây. Mạng nơ ron tích chập rất thành công trong việc thực thi các nhiệm vụ về thị giác máy yêu cầu tốc độ xử lý nhanh và độ chính xác cao.

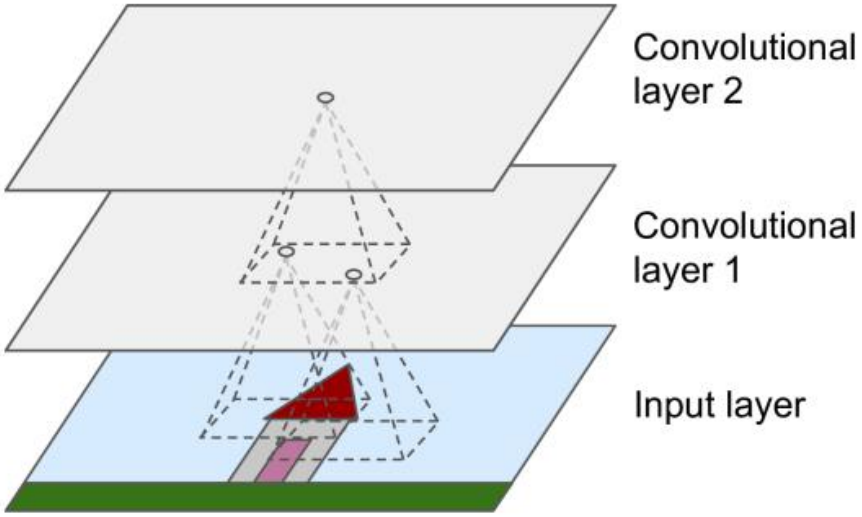
2.3.3.1. Khái niệm về mạng nơ ron tích chập

Mạng nơ ron tích chập là một trong những mạng truyền thẳng đặc biệt. Mạng nơron tích chập là một mô hình học sâu phổ biến và tiên tiến nhất hiện nay. Hầu hết các hệ thống nhận diện và xử lý ảnh hiện nay đều sử dụng mạng nơ ron tích chập vì tốc độ xử lý nhanh và độ chính xác cao. Trong mạng nơ ron truyền thống, các tầng được coi là một chiều, thì trong mạng nơ ron tích chập, các tầng được coi là 3 chiều, gồm: chiều cao, chiều rộng và chiều sâu. Mạng nơ ron tích chập có hai khái niệm quan trọng: kết nối cục bộ và chia sẻ tham số. Những khái niệm này góp phần giảm số lượng trọng số cần được huấn luyện, do đó tăng nhanh được tốc độ tính toán.

Trong các mục sau của phần 2.3.3. này, chúng ta sẽ đi tìm hiểu khái niệm về các lớp có trong một mạng nơ ron tích chập, từ đó hình dung ra quá trình hay những việc cần thiết để xây dựng một mạng nơ ron tích chập.

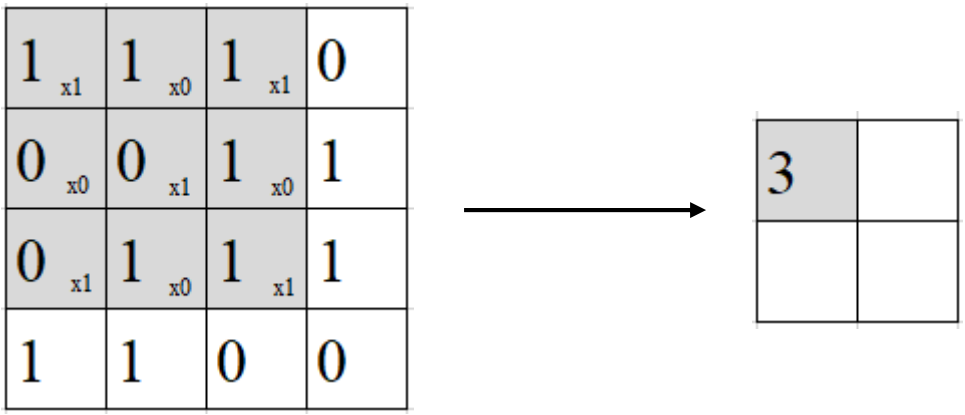
2.3.3.2. Lớp nơ ron tích chập – Convolutional Layer

Không giống như mạng nơ ron perceptron đơn thuần, ở mô hình perceptron, các nơ ron ở lớp ẩn đầu tiên sẽ nhận kết quả của các nơ ron đầu vào. Còn mô hình Convolutional layer, lớp tích chập đầu tiên chỉ kết nối với số lượng pixel có trong trường tiếp nhận.



Hình 2.8. Nơ ron tiếp nhận của lớp tích chập.

Trường tiếp nhận thực ra là sử dụng phép toán biến đổi số, phép toán biến đổi tích chập sử dụng một lớp ma trận để chuyển đổi đặc trưng của một ma trận khác lớn hơn. Hình phía dưới sẽ minh họa bằng ma trận tích chập có kích thước 3x3 mã hóa ma trận kích thước 4x4 với lần đầu tiên cho ra kết quả 3 ở ma trận 2x2.

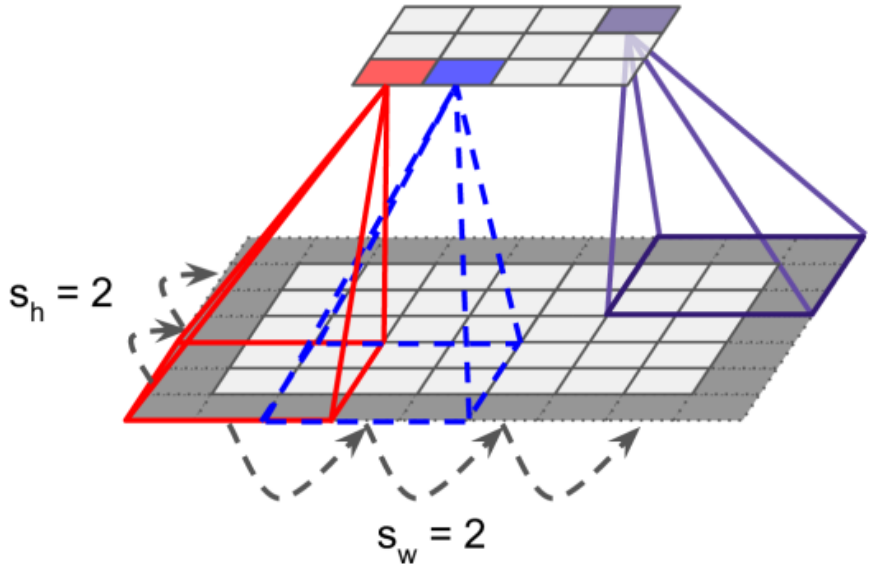


Hình ảnh Convolved feature

Hình 2.9. Một bước trong quá trình chuyển đổi bằng tích chập

Ma trận bên trái mà một hình ảnh đen trắng, với 1 biểu thị cho trắng, 0 biểu thị cho màu đen. Lớp ma trận 3x3 được tô màu xám còn được gọi là filter hay kernel. Giá trị 3 của ma trận Convolved feature là kết quả sau khi áp dụng tích chập.

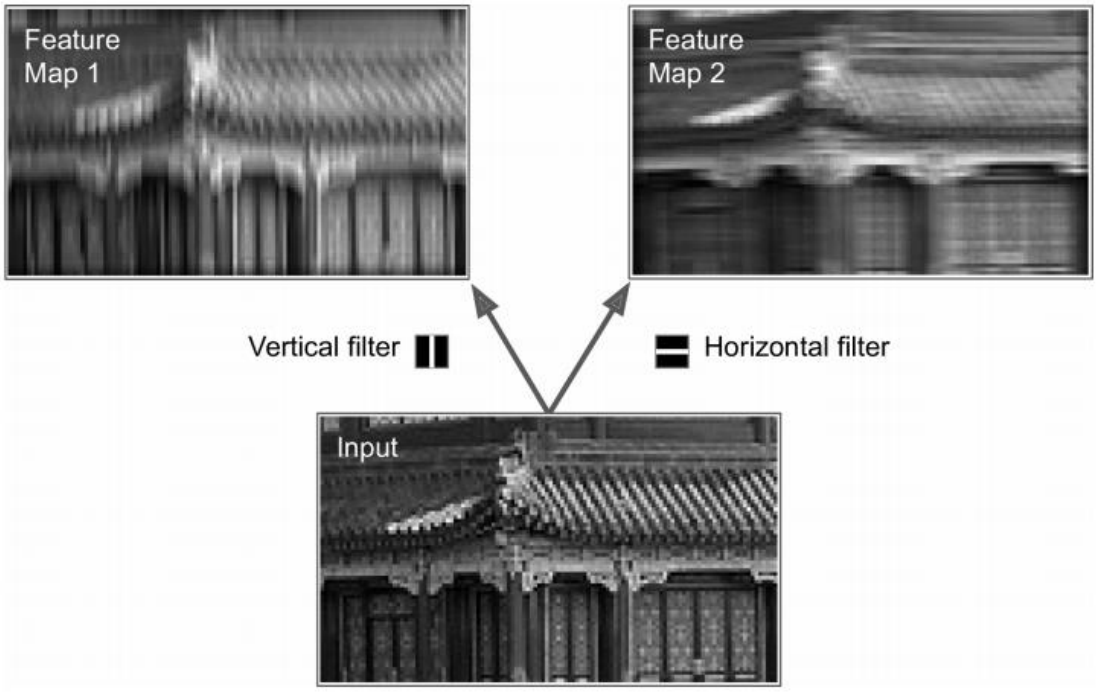
Tùy theo cách thuật toán di chuyển trường tiếp nhận mà trường tiếp nhận có thể vượt ra ngoài khung ảnh, từ đó khái niệm zero-padding ra đời, thuật toán sẽ tự gán cho các giá trị của trường tiếp nhận bị vượt ra khỏi khung ảnh thành giá trị 0. Điều đó k hề gây ảnh hưởng cho việc trích xuất đặc trưng của ảnh.



Hình 2.10. Zero-padding.

Hình 2.10 mô tả việc dịch trường tiếp nhận 2 pixel mỗi bước di chuyển (stride), và điều đó khiến trường tiếp nhận bị vượt ra khỏi khung ảnh thật.

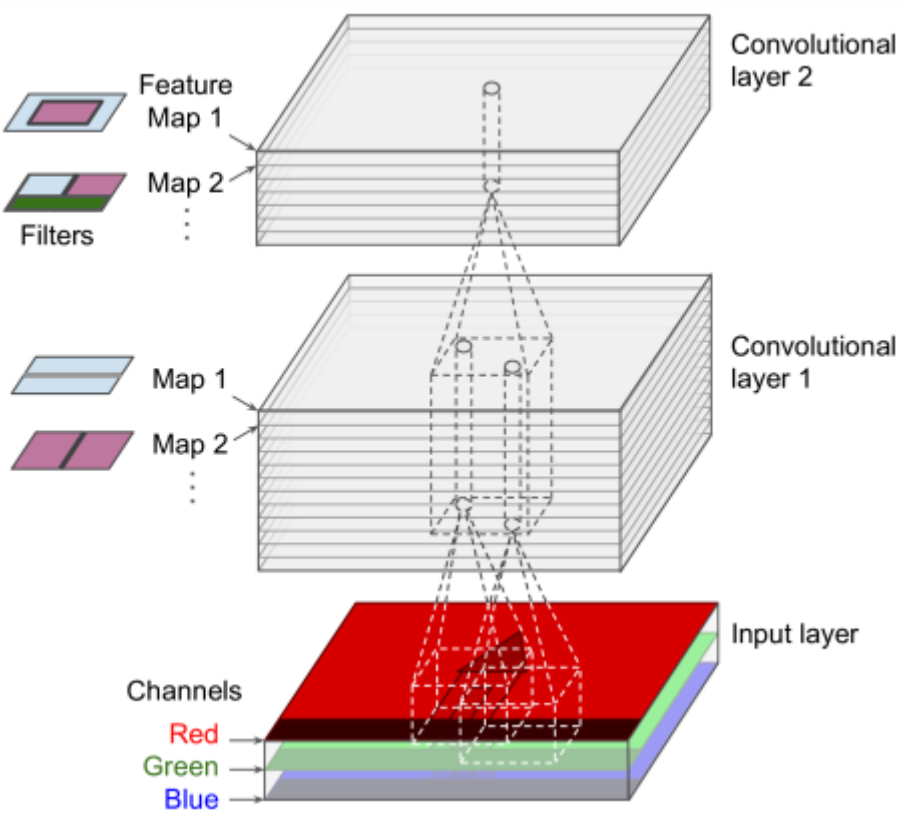
Sau đây là hình ảnh mô tả cho việc lấy đặc trưng hình ảnh từ hai lớp filter (trường tiếp nhận) khác nhau.



Hình 2.11. Lấy đặc trưng hình ảnh thông qua 2 lớp filter ngang và dọc.

Thuật toán được xây dựng sẵn từ các chuyên gia đã giúp cho máy có thể tự áp dụng các filter thích hợp cho từng trường hợp. Nhưng bạn vẫn có thể tự mình thiết kế filter và áp dụng vào trường hợp cụ thể mà bạn muốn, tất nhiên bạn phải nắm vững các kiến thức cần thiết. Và trên thực tế có đến vài trăm hoặc vài nghìn filters trong một mô hình phân tích hình ảnh.

Hình 2.11 cho chúng ta thấy khi áp dụng hợp lý các filters chúng ta sẽ có được các đặc trưng mà hình ảnh có. Từ đầu đến giờ chúng ta chỉ sử dụng các hình ảnh 2D hay các hình ảnh bao gồm 2 màu trắng và đen. Trong thực tế hiện nay, đã có một số công nghệ chụp vật thể 3D và hầu hết các hình ảnh được chụp dưới dạng RGB, RGB tức là Red – đỏ, Green – xanh lá, Blue – xanh da trời, là 3 màu cơ bản tổng hợp nên vô số các màu khác. Chính vì thế chúng ta phải áp dụng nhiều lớp filters để lấy được nhiều đặc trưng từ hình ảnh.

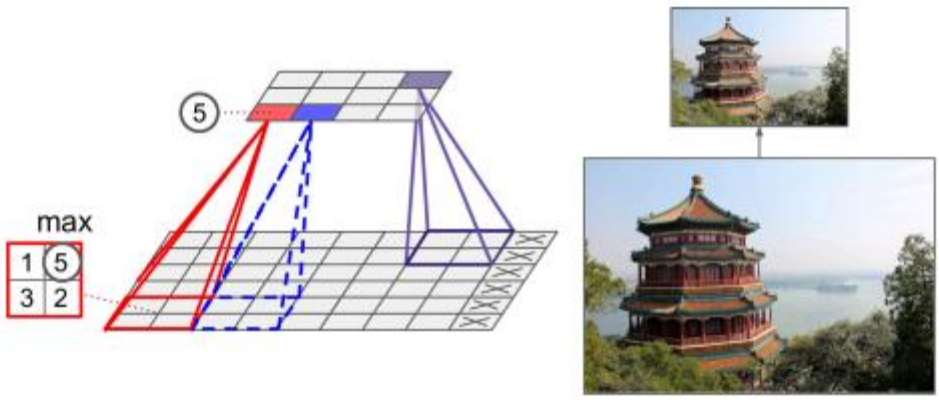


Hình 2.12. Các đặc trưng hình ảnh được lấy từ 1 ảnh dạng RGB.

2.3.3.3. Lớp gộp – Pooling Layer

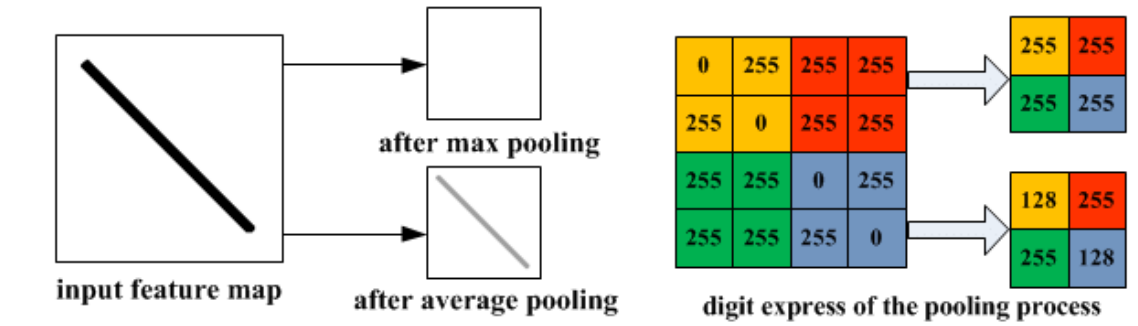
Khi bạn đã hiểu về các lớp tích chập, nội dung tiếp theo là lớp gộp (pooling layer). Pooling layer hiểu đơn giản là lớp thực hiện việc giảm tải lượng dữ liệu đầu vào, cụ thể hơn là thu nhỏ hình ảnh, giảm lượng pixel thể hiện đặc trưng hình ảnh đó sau mỗi lớp tích chập.

Hiện nay có vài phương thức để lấy giá trị đặc trưng bằng Pooling, chúng ta có thể kể đến là averagePooling – lấy giá trị trung bình, minPooling – lấy giá trị nhỏ nhất, maxPooling – lấy giá trị lớn nhất.

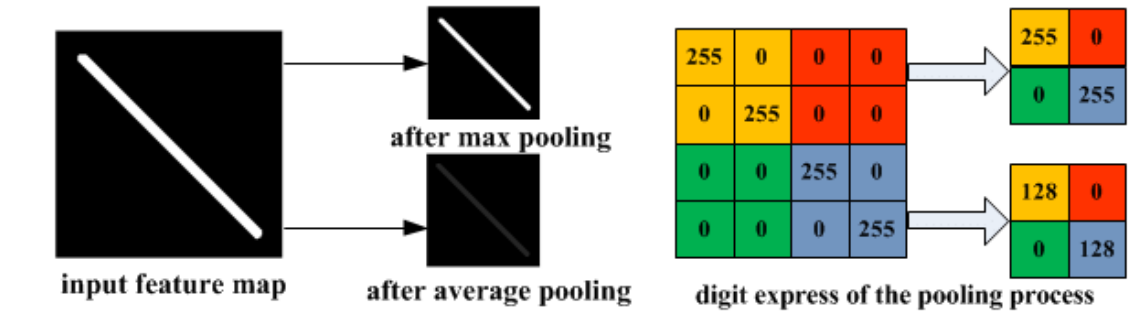


Hình 2.13. Áp dụng MaxPooling kích thước 2x2 bước nhảy 2 cho một ảnh RGB.

Tuy nhiên, Pooling layer vẫn có nhược điểm, đáng kể đến nhất là việc giảm kích thước hình ảnh sẽ gây ra sai sót, có thể một số đặc trưng hình ảnh sẽ bị bỏ qua một cách không mong muốn. Để dễ hình dung, chúng ta sử dụng MaxPooling kích thước 2x2 như hình 2.13, rõ ràng hình ảnh đầu ra sẽ nhỏ đi, diện tích bị giảm 75%, hình dưới đây sẽ minh họa cả MaxPooling và AveragePooling trong trường hợp gây ảnh hưởng đến đặc trưng hình ảnh.



(a) Illustration of max pooling drawback



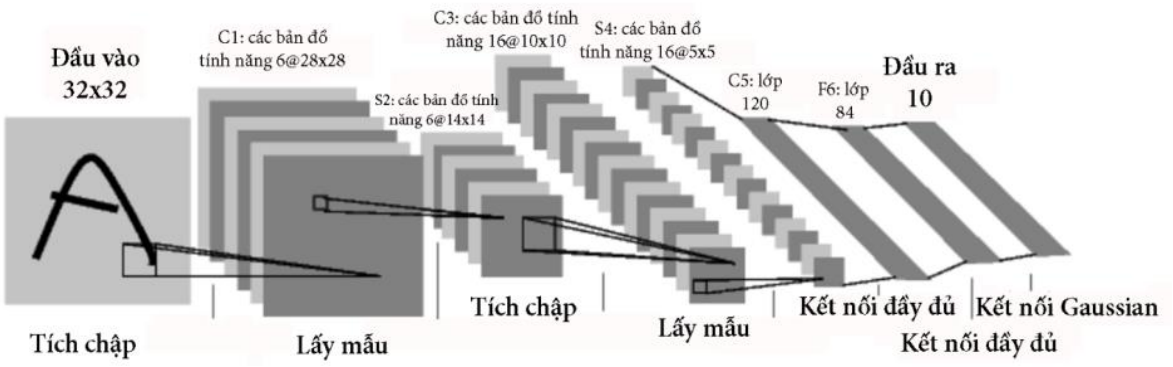
(b) Illustration of average pooling drawback

Hình 2.14. Nhược điểm của Pooling Layers.

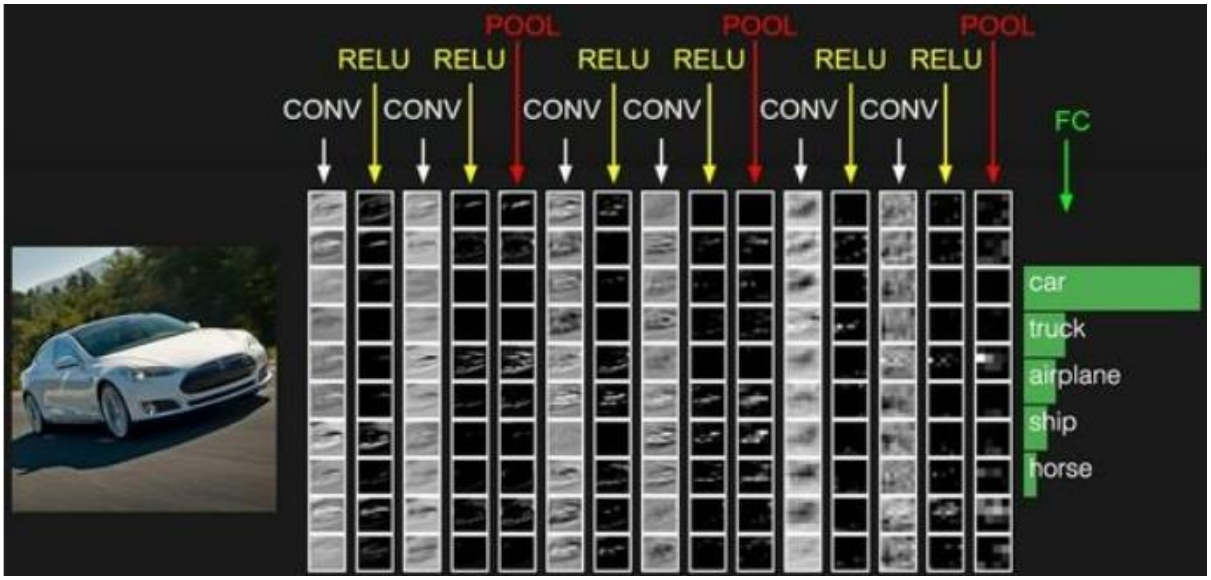
2.3.3.4. Xây dựng mạng nơ ron tích chập.

Để xây dựng một mạng nơ ron tích chập, chúng ta cần 3 tầng chính:

- Tầng tích chập (convolutional layer).
- Tầng gộp (pooling layer).
- Tầng được kết nối đầy đủ (fully-connected layer hoặc dense layer)



Hình 2.9. Mô hình minh họa mạng nơ ron tích chập – CNNs



Hình 2.10. Ví dụ minh họa sử dụng CNNs để phân loại vật.

Như chúng ta thấy trong Hình 2.10. một hình ảnh sẽ được áp dụng nhiều lớp filter khác nhau, thông thường sẽ có từ vài trăm đến vài nghìn filter như vậy. Mục đích nhằm trích xuất được các đặc trưng mà tấm ảnh có, nhằm lấy được tri thức chứa bên trong bức ảnh. Và đầu ra của layer trước sẽ là đầu vào cho layer sau, các layer sau cũng sẽ có lớp filter khác nhau.

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer như đã mô tả ở mục 2.3.3.3, Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs

cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên. Ta phân biệt được một con chó với một con mèo nhờ vào các đặc trưng từ mức độ thấp (có 4 chân, có đuôi) đến mức độ cao (dáng đi, hình thể, màu lông).

CHƯƠNG 3: DỮ LIỆU

3.1. Mô tả tập dữ liệu

Tập dữ liệu gồm 1981 hình ảnh người có đeo khẩu trang, 3318 hình ảnh người không có đeo khẩu trang. Tất cả được lấy ngẫu nhiên trên mạng.

Sau đó, chúng ta sẽ chia tập dữ liệu trên thành các training set và testing set. Trước tiên, chúng ta cần tạo đường dẫn lưu dữ liệu

```
MASK_SOURCE_DIR = "/content/drive/MyDrive/dataset/with_mask/"
TRAINING_MASK_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/training/mask/"
TESTING_MASK_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/testing/mask/"
WITHOUT_MASK_SOURCE_DIR = "/content/drive/MyDrive/dataset/without_mask/"
TRAINING_WITHOUT_MASK_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/training/nonmask/"
TESTING_WIHTOUT_MASK_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/testing/nonmask/"
```

Tạo hàm để chia tập dữ liệu thành training set và testing set. Với tỉ lệ chia là 9:1.

```
def split_data(SOURCE, TRAINING, TESTING, SPLIT_SIZE):
    files = []
    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " bị lỗi nên bỏ qua.")

    training_length = int(len(files) * SPLIT_SIZE)
    testing_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0:training_length]
    testing_set = shuffled_set[-testing_length:]

    for filename in training_set:
        this_file = SOURCE + filename
        destination = TRAINING + filename
        copyfile(this_file, destination)

    for filename in testing_set:
        this_file = SOURCE + filename
        destination = TESTING + filename
        copyfile(this_file, destination)

split_size = .9
split_data(MASK_SOURCE_DIR,
           TRAINING_MASK_DIR,
           TESTING_MASK_DIR,
           split_size)
split_data(WITHOUT_MASK_SOURCE_DIR,
           TRAINING_WITHOUT_MASK_DIR,
           TESTING_WIHTOUT_MASK_DIR,
           split_size)
```

Tạo tập dữ liệu train và tập dữ liệu validation (dùng để đánh giá kết quả học của máy). Lưu ý, chúng ta có dòng lệnh:

train_datagen = ImageDataGenerator(rescale = 1.0/255.0)

Mục đích là co giãn các giá trị của từng pixel bên trong ảnh, do ảnh thuộc loại RGB có 0-255 giá trị, chúng ta muốn co nó về sao cho các giá trị nằm trong khoảng (0,1).

Kèm theo class_mode = 'binary' nhằm xác định rằng trong tập dữ liệu đó được chia thành 2 class, mặc định ở đây là các giá trị 0, 1.

```
TRAINING_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/training/"
train_datagen = ImageDataGenerator(rescale=1.0/255.)
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=250,
                                                    class_mode='binary',
                                                    target_size=(150, 150))

VALIDATION_DIR = "/content/drive/MyDrive/dataset/mask_v_nonmask/testing/"
validation_datagen = ImageDataGenerator(rescale=1.0/255.)
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=250,
                                                              class_mode='binary',
                                                              target_size=(150, 150))
```

CHƯƠNG 4: GIẢI THUẬT VÀ CHƯƠNG TRÌNH

4.1. Chương trình và cách huấn luyện máy

Tạo mô hình mạng tích chập.

```
leaky_relu = keras.layers.LeakyReLU(alpha=0.2)

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation=leaky_relu,
                           input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation=leaky_relu),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation=leaky_relu),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=leaky_relu),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer=RMSprop(learning_rate=0.001),
              loss='binary_crossentropy', metrics=['acc'])
```

Mô hình CNNs ở trên có 9 lớp tất cả, bao gồm 3 lớp Convolutional nets với số nơ ron tăng dần là 16, 32 và 64. Xen kẽ giữa 3 lớp Convolutional nets là 3 lớp MaxPooling kích thước 2x2.

Tiếp đến là lớp Flatten, Flatten là một loại hoạt động reshape cụ thể trong đó tất cả các axes được làm phẳng (smooshed) hoặc ghép lại (squashed) với nhau. Hiểu đơn giản hơn, Flatten dùng để chuyển các ma trận kết quả có được từ hình ảnh sang dạng vecto.

Lớp kế tiếp là lớp kết nối đầy đủ, fully connected networks, hay còn gọi là Dense với 512 nơ ron, được xếp thành một vecto, tiếp nhận giá trị từ lớp Flatten ở phía trước. Cuối cùng lớp Dense một nơ ron để cho ra kết quả dự đoán của mô hình.

Tổng có 8 lớp sử dụng hàm kích hoạt leaky ReLU, nhằm tránh sai sót do hàm ReLU gây ra. Và lớp Dense cuối cùng sử dụng hàm kích hoạt Sigmoid.

Dòng lệnh cuối cùng, phải compile mô hình mạng nơ ron trước khi thực hiện việc train cho máy.

```
batch_size = 250
dataset_size = 3804
steps_per_epoch = int(0.75 * dataset_size / batch_size)
validation_steps = int(0.15 * dataset_size / batch_size)
history = model.fit(train_generator, epochs=30,
                    steps_per_epoch=steps_per_epoch,
                    validation_data=validation_generator,
                    validation_steps=validation_steps)
model.save('/content/drive/MyDrive/dataset/CNNs_face_detection_1.h5')
```

Sau đó, chúng ta thực hiện train cho máy. Với các thông số, một epoch được tính là khi chúng ta đưa tất cả dữ liệu vào mạng neural network 1 lần, khi mà dữ liệu quá lớn, chúng ta phải chia nhỏ nó ra thành batch_size. Ở trong phần tạo cơ sở dữ liệu, batch_size đã được gán giá trị là 250, nên chúng ta giữ nguyên.

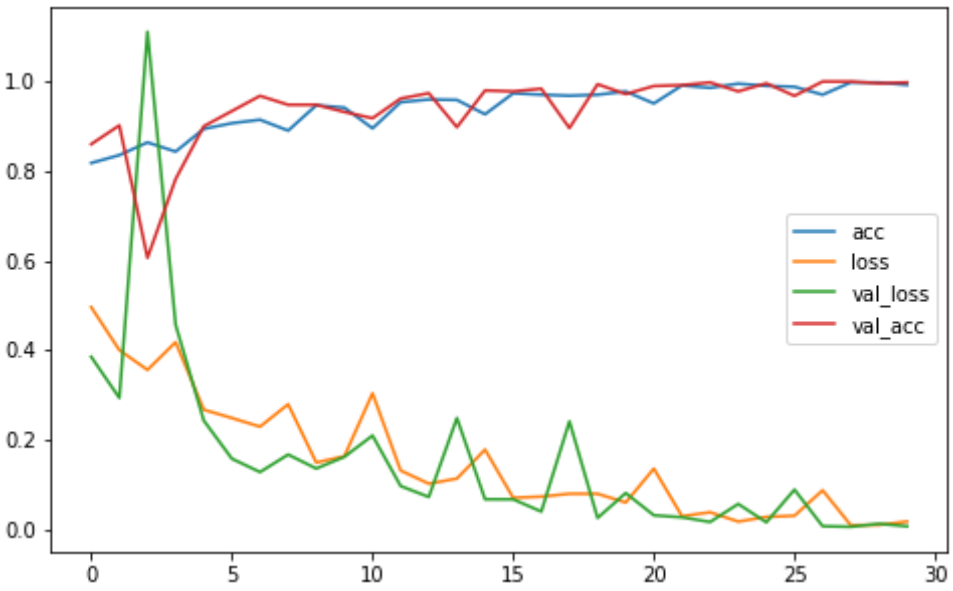
Còn dataset_size là kích thước của toàn bộ tập dữ liệu dùng để train. Để có được số bước thực hiện trên mỗi epoch chúng ta có công thức như trên. Và của số bước để đánh giá dữ liệu cũng đã có công thức.

Cuối cùng là lưu mô hình đã train được thành một file có định dạng h5fd, nhằm có thể sử dụng lại mô hình với tri thức đã được lưu bên trong, tránh mất thời gian để train lại mỗi khi mở máy lên.

4.2. Kết quả huấn luyện

Sau khi hoàn tất huấn luyện máy, các giá trị loss, acc, val_loss, val_acc sẽ được lưu lại trong biến history của mô hình. Chúng ta có thể gọi nó ra và vẽ đồ thị để xem xét chất lượng của mô hình mà chúng ta đã tạo.

```
# pd.DataFrame(history.history['acc','loss']).plot(figsize=(8, 5))
data = {}
data['acc'] = history.history['acc']
data['loss'] = history.history['loss']
data['val_loss'] = history.history['val_loss']
data['val_acc'] = history.history['val_acc']
pd.DataFrame(data).plot(figsize=(8, 5))
```



CHƯƠNG 5: KẾT LUẬN

5.1. Kết quả cuối cùng

Chúng ta sẽ tạo hàm để có thể chụp ảnh trực tiếp trên google colab.

```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

Để kiểm tra mô hình chúng ta tạo hàm:

```
classified_dict = {}
mask = 0
nonmask = 0
checkmodel_dir = '/content/drive/MyDrive/dataset/face_with_mask/'
for filename in os.listdir(checkmodel_dir):
    img_dir = checkmodel_dir + filename
    images = image.load_img(img_dir, target_size=(150, 150))
    x = image.img_to_array(images)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    if classes[0]>0.4:
      text = "without mask"
      nonmask += 1
    else:
      text = " with mask"
      mask += 1
    classified_dict[img_dir] = text
print("mask: ", mask, "\n nonmask: ", nonmask)
print(classified_dict)
```

Chúng ta lần lượt sử dụng nhiều hình ảnh khác nhau. Đầu tiên là 138 ảnh RGB khuôn mặt người có đeo khẩu trang, sau đó là bộ dữ liệu gồm 200 ảnh RGB khuôn mặt người không đeo khẩu trang. Độ chính xác đạt được sau khi chạy thử 322 ảnh trên là 89.4%. Như vậy kết quả của model không được tốt lắm so với các mô hình phát hiện khuôn mặt phổ biến trên internet hiện nay.

5.2. Các đề xuất cải tiến tiếp theo

Để tăng độ chính xác của mô hình, chúng ta cần thêm một số câu lệnh tính toán để loại bớt một số sai sót như overfitting. Ở trường hợp mô hình nhận diện khuôn mặt đeo khẩu trang hiện tại có 2 vấn đề. Đầu tiên là tập dữ liệu khá nhỏ, khoảng hơn 4000 hình ảnh, nên chúng ta cần chia nhỏ (batch_size) giảm số hình ảnh đưa vào trong một lượt.

Sau đó là thêm hàm Batch Normalization để tránh overfitting.

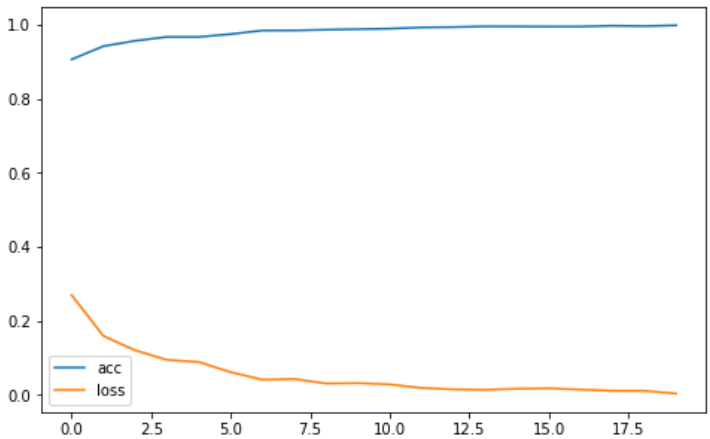
```
[3]  model = tf.keras.models.Sequential([
         tf.keras.layers.Conv2D(16, (3, 3), activation='relu',
                                input_shape=(150, 150, 3)),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(512, activation='relu'),
         tf.keras.layers.BatchNormalization(),
         tf.keras.layers.Dense(1, activation='sigmoid')
     ])

     model.compile(optimizer=RMSprop(learning_rate = 0.001),
                   loss='binary_crossentropy', metrics=['acc'])
```

Chúng ta thêm lớp tf.keras.layers.BatchNormalization() trước lớp Dense cuối cùng để loại bỏ các giá trị overfitting.

Và chúng ta sẽ giảm batch_size còn 47, so với lúc đầu là batch_size = 250 trong khi dataset_size = 3804.

```
batch_size = 47
dataset_size = 4768
steps_per_epoch = int(0.75 * dataset_size / batch_size)
validation_steps = int(0.15 * dataset_size / batch_size)
history = model.fit(train_generator, epochs=20,
                    steps_per_epoch=steps_per_epoch,
                    validation_data=validation_generator,
                    validation_steps=validation_steps)
model.save('/content/drive/MyDrive/dataset/CNNs_face_detection_model_tflite.h5')
```

Sau khi train model chúng ta có đồ thị sau:



Có thể thấy rằng độ chính xác và giá trị mất mát hội tụ nhanh hơn và không có nhiễu.

Kiểm tra model sau khi cải thiện bằng tập dữ liệu gồm 500 ảnh khuôn mặt người không đeo khẩu trang và 500 ảnh người có đeo khẩu trang. Kết quả đạt được là 95.6%. Độ chính xác đã được cải thiện rõ rệt. Tuy nhiên, muốn cải thiện thêm chúng ta cần xem xét và thay đổi các hàm kích hoạt, tăng hoặc giảm nơ ron hoặc các lớp một cách cẩn thận.

TÀI LIỆU THAM KHẢO:

[1] Lê Thị Thu Hằng, "Nghiên cứu về mạng neural tích chập và ứng dụng cho bài toán nhận diện biển số xe", Đại học quốc gia Hà Nội, Trường đại học công nghệ, Hà Nội, 2016

[2] Nguyễn Mạnh Hùng, "Nghiên cứu về mạng neural convolutional, áp dụng vào bài toán nhận dạng đối tượng trong lĩnh vực thị giác máy tính" , Đại học quốc gia Hà Nội, Trường đại học công nghệ, Hà Nội, 2019

[3] Asit Kumar Datta, Madhura Datta, Pradipta Kumar Banerjee, "Face Detection and Face Recognition – Theory and Practice".

[4] Aurelien Geron, "Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow."

[5] Neural Network-Based Face Detection, by Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[6] Code mẫu trên Kaggle:

https://www.kaggle.com/rawanmajedlazkani/mask-social-distancing-detection-using-mobilenet

[7] Code mẫu trên google colab:

https://colab.research.google.com/github/lmoroney/mlday-tokyo/blob/master/Lab6-Cats-v-Dogs.ipynb#scrollTo=LqL6FYUrtXpf