ĐẠI HỌC BÁCH KHOA HÀ NỘI

PROJECT 1

MÔ HÌNH CONVOLUTION NEURAL NETWORK VÀ PHÁT TRIỂN ỨNG DỤNG NHẬN DIỆN VIỆC ĐEO KHẨU TRANG TRÊN KHUÔN MẶT NGƯỜI DÙNG

NGUYỄN THANH TÙNG

Tung.NT229125@sis.hust.edu.vn

Ngành Công nghệ Thông tin

Giáo viên hướng dẫn	:	THS. Nguyễn Đức Tiến
Bộ môn	:	Kỹ thuật Máy tính
Viện	:	Công nghệ thông tin và Truyền thông

Hà Nội, 11/2023

Lời nói đầu

Trong thời đại ngày nay, khi công nghệ ngày càng phát triển, việc nghiên cứu và ứng dụng trí tuệ nhân tạo đang trở thành một phần quan trọng của nhiều lĩnh vực trong đời sống xã hội. Dự án mà em trình bày dưới đây không chỉ là một nỗ lực hướng đến sự đổi mới trong lĩnh vực nhận diện khuôn mặt, mà còn là sự kết hợp giữa sự hiểu biết sâu rộng về Convolutional Neural Networks (CNN) và một ứng dụng thực tế hữu ích - phần mềm nhận diện khuôn mặt với khả năng phân biệt việc đeo khẩu trang.

Em đã tập trung vào việc áp dụng công nghệ học máy, đặc biệt là CNN, để xây dựng một mô hình nhận diện khuôn mặt chính xác và linh hoạt. Điều này không chỉ mang lại những tiện ích trong lĩnh vực an ninh và giám sát mà còn đáp ứng một nhu cầu ngày càng tăng của cộng đồng - kiểm soát việc đeo khẩu trang để đảm bảo an toàn y tế.

Em xin chia sẻ những khám phá, thách thức, và kết quả của dự án này thông qua các phần tiếp theo của bản báo cáo. Hy vọng rằng nó sẽ là một hành trình thú vị và có ý nghĩa, đồng thời làm phong phú thêm hiểu biết về ứng dụng của học máy trong giải quyết các vấn đề thực tế.

Chân thành cảm ơn sự quan tâm và hỗ trợ từ phía Thầy Nguyễn Đức Tiến trong suốt quá trình thực hiện dự án này.

Sinh viên thực hiện

Nguyễn Thanh Tùng

Mục lục

Danh mục hình ảnh

Chương I. Tổng quan về đề tài

1. Giới thiệu bài toán

Trong thời đại hiện đại, sự tiến bộ của khoa học công nghệ, đặc biệt là trí tuệ nhân tạo (AI), đã tạo ra những đổi mới mạnh mẽ trong nhiều lĩnh vực cuộc sống. Áp dụng công nghệ và trí tuệ nhân tạo không chỉ giúp cải thiện hiệu suất công việc mà còn mở ra những khả năng mới, từ tự động hóa sản xuất đến phân tích dữ liệu phức tạp.Ứng dụng

2. Mục tiêu chính

Mục tiêu chính của đề tài là xây dựng một mô hình CNN hiệu quả để nhận diện việc đeo khẩu trang trên khuôn mặt người dùng. Qua đó, chúng ta mong muốn cung cấp một công cụ hữu ích hỗ trợ trong việc duy trì biện pháp phòng ngừa như đeo khẩu trang, đặc biệt trong những tình huống đặc biệt như đại dịch.

3. Phương pháp phát triển

- Thu Thập và Tiền Xử Lý Dữ Liệu
    - Thu thập dữ liệu đa dạng và thực tế.
    - Tiền xử lý dữ liệu với chuẩn hóa, cắt ghép, và augmentations để tăng khả năng tổng quát hóa của mô hình.
- Xây Dựng và Huấn Luyện Mô Hình
    - Xây dựng mô hình CNN và huấn luyện với các tham số như learning rate, số epoch, và batch size được điều chỉnh.
- Đánh Giá và Tinh Chỉnh
    - Đánh giá mô hình bằng các độ đo như độ chính xác, độ nhạy, và độ đặc biệt.
    - Tinh chỉnh mô hình để cải thiện khả năng nhận diện và giảm sai sót.
- Triển khai ứng dụng
    - Triển khai mô hình vào ứng dụng nhận diện đeo khẩu trang trên khuôn mặt, có thể tích hợp vào ứng dụng di động hoặc hệ thống an ninh tự động.

Chương II. Cơ sở lý thuyết

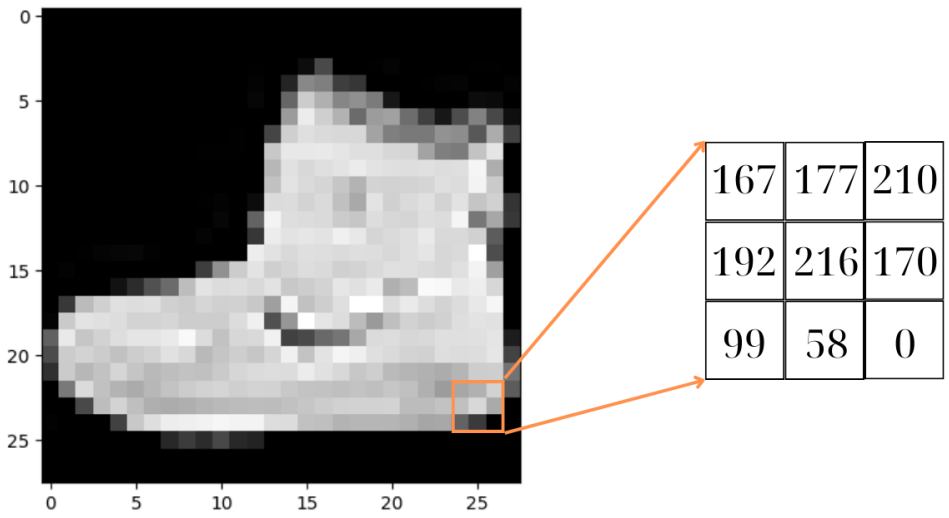
1. Hình ảnh là gì?

Hình ảnh là một khái niệm quen thuộc đối với mỗi người sống hiện nay, Hình ảnh có thể có hai chiều, như thể hiện trên tranh vẽ trên mặt phẳng, hoặc ba chiều, như thể hiện trên tác phẩm điêu khắc hoặc hologram. Hình ảnh có thể được ghi lại bằng thiết bị quang học như máy ảnh, gương, thấu kính, kính viễn vọng, kính hiển vi do con người tạo ra, hoặc bởi các cơ chế tự nhiên, như mắt người hay mặt nước. Hình ảnh có thể được dùng theo nghĩa rộng, thể hiện bản đồ, đồ thị, nghệ thuật trừu tượng. Hình ảnh còn có ý nghĩa đối với trí nhớ, tâm trí và văn hoá nhân loại.

Nhưng khi nói về xử lý ảnh ta có thể hiểu:

a. Ảnh là một ma trận

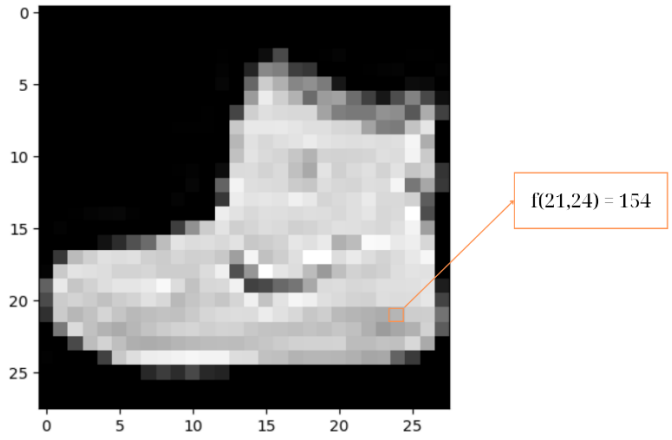
Ảnh được cấu tạo từ các pixel và mỗi ô pixel là một giá trị biểu thị cho độ sáng tối của bức ảnh đó, ví dụ như hình 1.1 mô tả mỗi pixel chứa một giá trị biểu thị 0 là tối nhất và 255 là sáng nhất. Khi hiển thị ở dạng số ta có thể thấy ảnh là một ma trận



Hình II-1 Mô tả cấu tạo của ảnh

b. Ảnh là một hàm

Như đã nói, ảnh là ma trận các pixel mang giá trị từ 0 đến 255. Ngoài ra ảnh có thể được hiểu như một hàm $f(x,y)$ với y là chiều cao và x là chiều ngang của ảnh. Đối với ảnh màu, ảnh có thêm 1 chiều là kênh (channel). Có 3 kênh trong ảnh màu và từng kênh biểu thị cho từng màu đỏ, xanh lục, xanh dương trong ảnh.



Hình II-2 Ảnh là một hàm

2. Gradient Descent

Hiện nay, Gradient descent là một trong những thuật toán phổ biến nhất để thực hiện tối ưu hóa và là cách phổ biến nhất để tối ưu hóa mạng nơ-ron.

Ý tưởng của thuật toán là việc thực hiện cập nhật điều chỉnh các tham số θ của mô hình $h_\theta(x)$ theo hướng giảm độ lớn của đạo hàm (gradient) của hàm mất mát $J(\theta)$ tại điểm hiện tại. Với $J(\theta)$ là hàm đo lường sai lệch của giá trị dự đoán với giá trị mục tiêu của mô hình. Hàm mất mát $J(\theta)$ trong có rất nhiều biến thể nhưng trong trường hợp này ta sử dụng MAE (Mean absolute error)

$$J(\theta) = \sum |\hat{y} - y|$$

Cụ thể: Ta có một mảng dữ liệu đầu vào x = [1,3,5,7] và mảng dữ liệu đầu ra là y = [2,4,6,8] ta phải tìm θ_0, θ_1 sao cho hàm $h_\theta(x)$ sấp sỉ với y:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Mục tiêu của bài toán là tối thiểu hoá $J(\theta)$ với

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} |h_\theta(x) - y|$$

Để thực hiện mục tiêu trên ta thực hiện như sau:

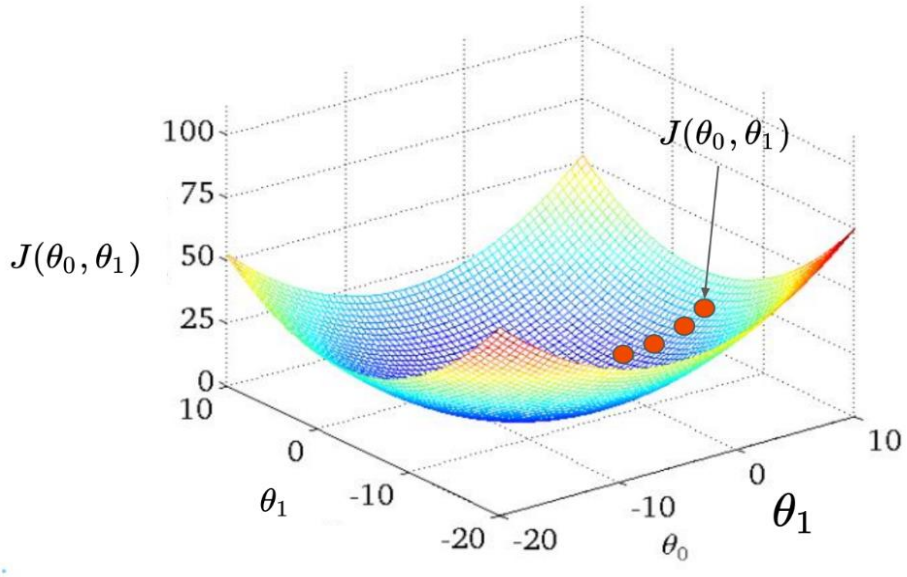
Bước 1: khởi tạo θ_0, θ_1với giá trị bất kỳ

Bước 2: thay đổi giá trị θ_0, θ_1 để $J(\theta)$ giảm dần cho đến giá trị cực tiểu

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j}$$

Với j = {0, 1}

Ta có thể thấy giá trị $J(\theta)$ giảm dần về cực tiểu ở hình



Hình II-3 Mô phỏng thuật toán GD

3. Softmax – Bài toán phân loại nhiều lớp

Ta đã được làm quen với hai khái niệm quan trọng trong học máy là hàm mục tiêu và hàm mất mát. Đối với ví dụ trên, hàm mục tiêu là: $h_\theta(x) = \theta_0 + \theta_1 x$ (Linear Regression) và hàm mất mát là MAE. Nhưng hàm mục tiêu trên chỉ phù hợp với bài toán dự đoán (ví dụ ta cho số km đi cần đi khi đi taxi và mô hình sẽ dự đoán số tiền phải trả). Còn với bài toán phân loại ta cần phân biệt đầu vào là thuộc nhãn nào trong các nhãn cần phân biệt. Ví dụ ta có bài toán phân biệt các ảnh [chó, mèo, ngựa]. Giả sử ta quy định các nhãn là [0,1,2] thì khi cho ảnh chó mô hình sẽ cho kết quả là 1, mèo là 2,… ta có thể thấy rằng không thể dùng Linear regression trong trường hợp này vì giả sử ta có 2 ảnh chó khác nhau, nếu cho vào mô hình dùng Linear regression thì 2 ảnh này cho ra hai giá trị mặc dù có thể gần nhau nhưng vẫn khác nhau và thứ ta cần là kết quả của hai ảnh đó đều phải bằng 0. Nói cách khác Linear regression dự đoán giá trị liên tục trong khi ta đang cần một mô hình có thể dự đoán giá trị rời rạc.

a. Hàm Softmax

$$h_\theta(x)_k = \frac{e^{(z_k)}}{\sum_{t=1}^{c} e^{(z_k)}}$$

c: $số\ nhãn$

k: $nhãn\ hiện\ tại$

Ví dụ ta có mảng [5,2,4,3] thì

$$h_\theta(x)_1 = \frac{e^5}{e^5 + e^2 + e^4 + e^3} = 0.64$$

Tương tự ta có với từng phần tử trong mảng ta có [0.64, 0.03, 0.23, 0.08]. Ta có thể thấy kết quả cho ra thể hiện cho xác suất của từng nhãn và thể hiện tính chất xác suất khi tổng các xác suất bằng 1. Ta cũng có thể thấy hàm softmax cực đại hoá xác suất ở phần tử có giá trị lớn. Đây là một trong những tính chất của hàm softmax

Sâu hơn về softmax, ta có bài toán phân loại ba nhãn [1,2,3] và ta có 1 điểm dữ liệu x và ta cần phải dự đoán x thuộc về nhãn nào trong ba nhãn. Giả sử:

$$P(y=1|x) = h_\theta(x)_1 = softmax(z_1) = 0.7$$
$$P(y=2|x) = h_\theta(x)_2 = softmax(z_2) = 0.2$$
$$P(y=3|x) = h_\theta(x)_3 = softmax(z_3) = 0.1$$
$$trong \text{ đó: } z_i \text{ là đầu } ra \text{ của } hàm \text{ } hàm \text{ } phi \text{ } tuyến$$

Ta có thể thấy kết quả mô hình dự đoán x có nhãn là 1 là 0.7. Nhưng chưa chắc 1 là nhãn đúng của x. vậy có cách nào để lấy là kết quả mô hình với nhãn thật của điểm dữ liệu. Ta có công thức sau:

$$P(y|x) = h_\theta(x)_1^{1\{y=1\}}.h_\theta(x)_2^{1\{y=2\}} \ldots . h_\theta(x)_c^{1\{y=c\}}$$
$$= \prod_{k=1}^{c} h_\theta(x)_k^{1\{y=k\}}$$

Giải thích: {y=c} ở đây có thể hiểu là một phép toán logic, với nhãn của y = 2 ta có {y=2} = 1, {y=1} = 0. Vậy với y = 2 ta có:

$$P(y|x) = h_\theta(x)_1^0 \,.\, h_\theta(x)_2^1 \,.\, h_\theta(x)_3^0 = h_\theta(x)_2 = 0.2$$

b. Hàm mất mát

Như đã nói, đối với bài toán phân loại nhiều lớp, ta không thể dùng hàm mất mát là MAE. Vậy có hàm mất mát nào giúp ta tối ưu kết quả của bài toán.

Dưới đây là công thức tính xác suất đầu ra của mô hình trên một điểm dữ liệu với nhãn đúng. Vậy ta mong muốn điều gì ở kết quả của công thức đó? Có phải đầu ra của mô hình đối với nhãn đúng phải là cao nhất có thể? Vậy nhiệm vụ của ta là cực đại hoá P(y|x).

$$P(y|x) = \prod_{k=1}^{c} h_\theta(x)_k^{1\{y=k\}}$$

Tổng quát hơn với 1 bộ dữ liệu ta cũng muốn xác suất đúng của tất cả điểm dữ liệu là cao nhất có thể. Áp dụng công thức xác suất ta có:

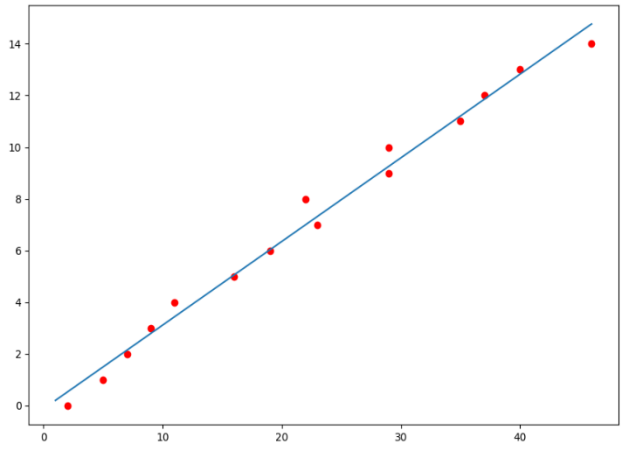
$$L(\theta) = \prod_{i=1}^{m} \prod_{k=1}^{c} h_\theta(x)_k^{1\{y=k\}}$$

Log hai vế ta được:

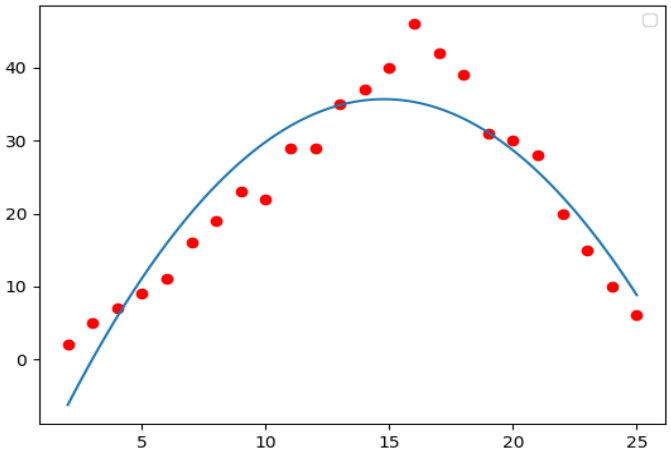
$$J(\theta) = -LL(\theta) = \sum_{i=1}^{m}\sum_{k=1}^{c} 1\{y = k\} log(\, h_\theta(x^{(i)})_k)$$

4. Mạng Nơ – ron

Trong phần trước ta đã tìm hiểu về mô hình dự đoán Linear Regression. Như hình dưới, ta thấy cách hoạt động của mô hình là khớp một phương trình vào dữ liệu (ở hình bên trái là đường thẳng) sao cho trung bình khoảng cách (MAE) là nhỏ nhất. Hay ở hình dưới là ví dụ cho hàm mục tiêu là hàm bậc 2:

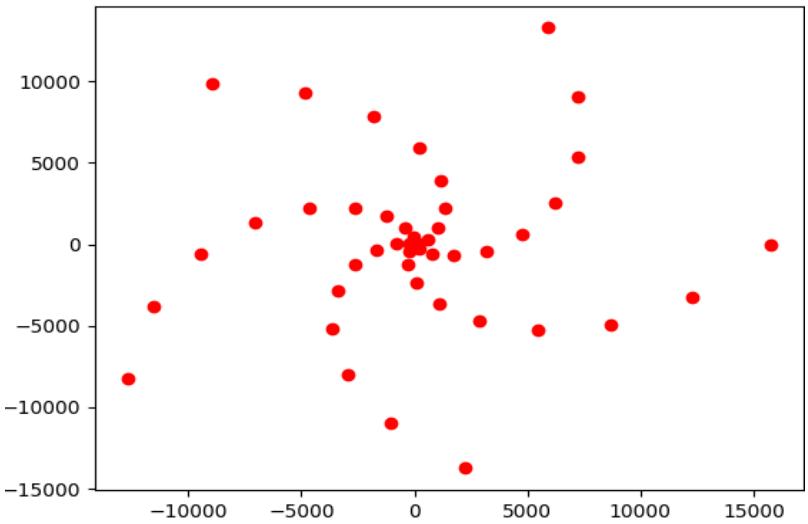


Hình II-4Linear regression với đường thẳng



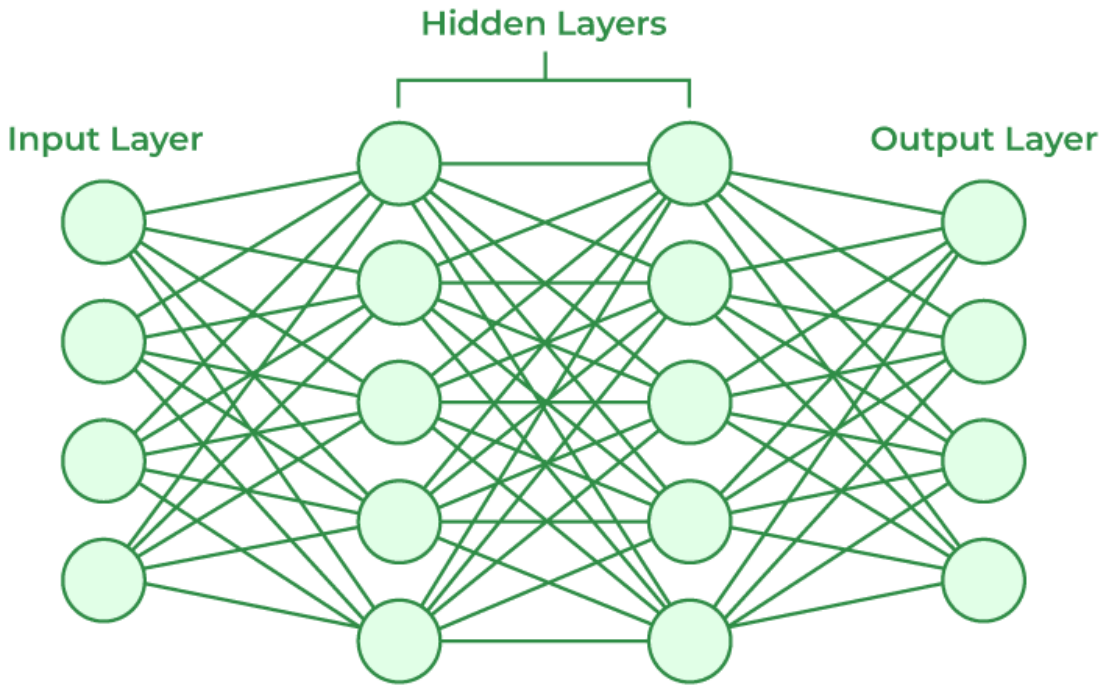
Hình II-5Liner regression với đường cong

Nhưng đối với hình dưới thì ta có thể có phương trình này khớp được với dữ liệu như vậy? Có một cách dễ hơn đó là sử dụng mạng nơ – ron.



a. Cấu tạo mạng nơ – ron

Mạng nơ-ron bao gồm một số lớp (layers) của các nơ-ron nhân tạo, mỗi nơ-ron là một đơn vị tính toán cơ bản. Mỗi nơ-ron nhận đầu vào từ nơ-ron trước đó, thực hiện các phép toán trên dữ liệu đầu vào, và sau đó truyền kết quả đến nơ-ron tiếp theo. Mạng nơ – ron đặc biệt ở việc nó được cấu tạo gồm các lớp: đầu vào, ẩn, đầu ra và chính những lớp ẩn giúp mạng nơ - ron tăng độ phực tạp của hàm mục tiêu lên nhiều lần mà không cần đến sự kiểm soát của con người.



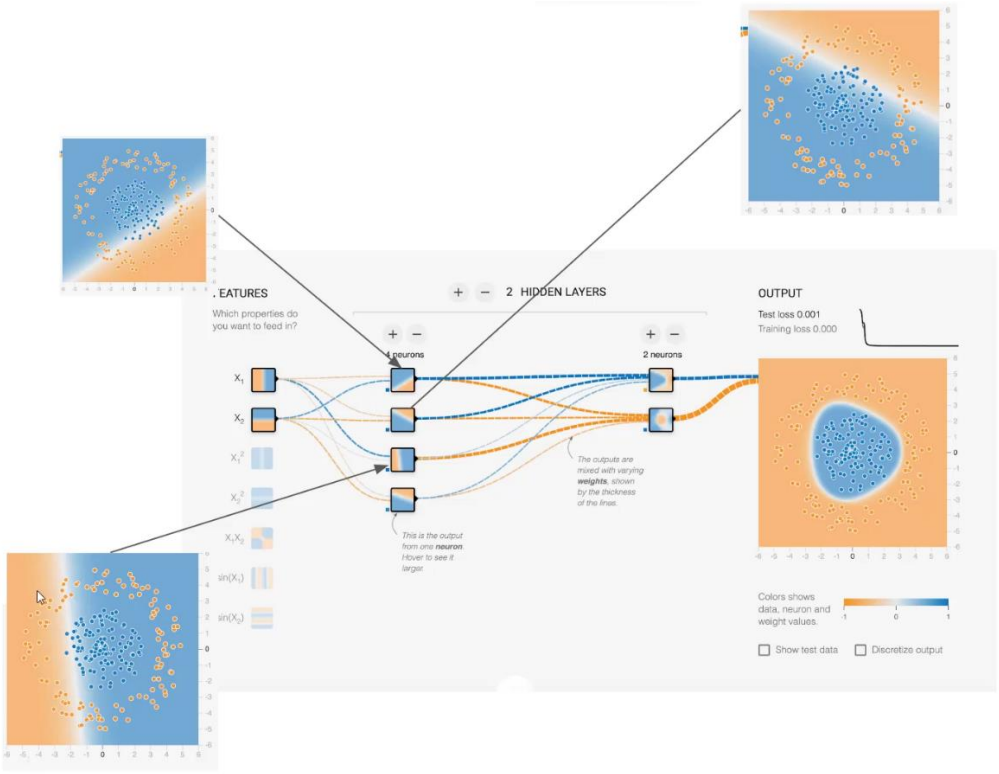
Hình II-6Cấu trúc mạng nơ-ron-1

b. Lớp ẩn và nơ – ron

Chúng ta có thể xem mạng nơ – ron như một nhà máy sản xuất với từng đường thẳng nối các nơ – ron và các nơ - ron là từng công đoạn. Mỗi đường thẳng có nhiệm vụ chế biến nguyên liệu trong khi mỗi nơ ron có nhiệm vụ lọc đi những nguyên liệu không cần thiết, hư hỏng,… và giữ lại những nguyên liệu cần thiết, hữu dụng cho lớp sau.

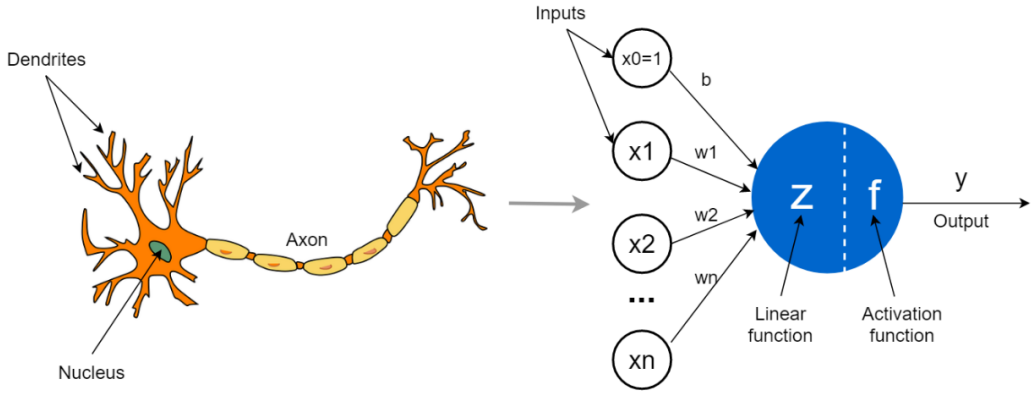
Điểm đặc biệt của mạng nơ - ron là từng nơ - ron, nơi có thể lọc đi và giữ lại những thông tin cần thiết và không cần thiết của dữ liệu, nói cách khác mạng nơ – ron có thể làm giàu dữ liệu.

Hình dưới mô mỏng lại quá trình phân loại. ta có thể thấy mạng có 1 lớp ẩn mỗi nơ – ron trong lớp ẩn lại có một đặc điểm khác nhau và mỗi lớp ẩn lại được lọc đi hay dữ lại dữ liệu thể hiện ở độ đậm nhạt của đường nối với lớp sau. Cuối cùng khi kết hợp lại với nhau thì ta được một phân loại cực tốt mà khó thể nào tinh chỉnh được với linear regression.



Hình II-7Cấu trúc mạng nơ ron-2

Vậy cùng xem tại sao các nơ – ron lại có thể trích xuất được các đặc điểm tốt và loại bỏ đi các đặc điểm thừa của dữ liệu



Hình II-8Cấu trúc mạng nơ ron-3

Sâu hơn vào từng nơ – ron, ta có thể thấy nơ – ron được cấu tạo từ hai phần là Linear function và Activation function.

- Linear function

Linear function hiểu đơn giản là Linear Regression nhưng thay θ bằng ma trận W.

$$Z = X^T \cdot W + b$$

- Activation function

Activation funtion hay còn gọi là phi tuyến là điểm đặc biệt của mạng nơ – ron nơi các đặc điểm hữu ích được giữ lại và loại bỏ đi các đặc điểm không cần thiết. Phi tuyến có rất nhiều biến thể có thể kể đến như:

- Relu

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & otherwise \end{cases} \; hay \; f(x) = max(x, 0)$$

- $Tanh$

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

5. Thuật toán lan truyền ngược trong mạng nơ – ron

a. Mục tiêu của thuật toán lan truyền ngược

Như đã tìm hiểu ở Gradient Descent, mục tiêu của ta là điều chỉnh các tham số của mô hình theo hướng giảm độ lớn của đạo hàm (gradient) của hàm mất mát $J(\theta)$ tại điểm hiện tại. Đối với mạng nơ – ron, mục tiêu của ta cũng như vậy nhưng ta phải tính đạo hàm của tham số với giá trị mất mát của từng lớp trong mạng.

Các tham số của mạng nơ-ron đơn giản với một tầng ẩn là $W^{(l)}$. Mục đích của lan truyền ngược là để tính gradient $\frac{\partial J}{W^{(l)}}$. Để làm được điều này, ta áp dụng quy tắc dây chuyền và lần lượt tính gradient của các biến trung gian và tham số. Các phép tính trong lan truyền ngược có thứ tự ngược lại so với các phép tính trong lan truyền xuôi.

b. Công thức

Bỏ qua phần chứng mình, ta có công thức tính tham số trên từng lớp lớp như sau:

- Ở lớp cuối cùng:

$$E^{(L)} = \frac{\partial J(W)}{\partial Z^{(L)}} softmax\left(Z^{(L)}\right) - Y$$

- Ở lớp ẩn:

$$dA^{(l-1)} = \frac{\partial J(W)}{\partial A^{(l-1)}} = W^{(l)T} dZ^{(l)} = W^{(l)T} E^{(l)}$$

$$E^{(l)} = dZ^{(l)} = dA^{(l)} * f'(Z^{(l)})$$

- Cập nhật tham số:

$$dW^{(l)} = \frac{\partial J}{W^{(l)}} = \frac{1}{m} dZ^{(l)} A^{(l-1)T} = \frac{1}{m} E^{(l)} A^{(l-1)T}$$

$$\rightarrow W^{(l)} := W^{(l)} - \alpha \frac{\partial J(W)}{\partial W^{(l)}} = W^{(l)} - \alpha \frac{1}{m} E^{(l)} A^{(l-1)T}$$

$$db^{(l)} = \frac{\partial J}{b^{(l)}} = \frac{1}{m}\sum_{i=1}^{m} dZ^{(l)(i)} = \frac{1}{m}\sum_{i=1}^{m} E^{(l)(i)}$$

$$\rightarrow b^{(l)} \coloneqq b^{(l)} - \alpha \frac{\partial J(b)}{\partial b^{(l)}} = b^{(l)} - \alpha \frac{1}{m}\sum_{i=1}^{m} E^{(l)(i)}$$

6. Mạng tích chập – Convolution neural network

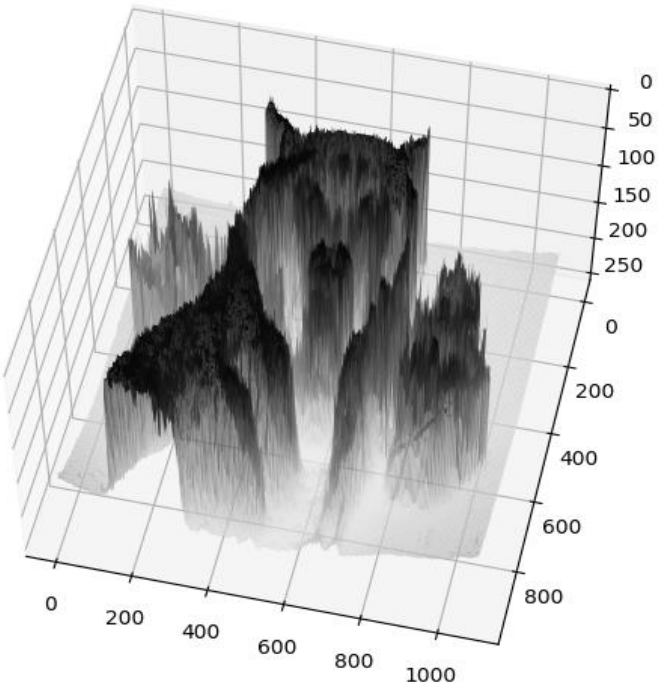
a. Phát hiện cạnh

Trong thực tế để nhận diện được một vật thể ta sẽ thường quan tâm để hình dáng của nó, và hình dáng của vật thể đó được định hình khi mà ta nhận ra được viền của vật thể đó.

Ví dụ ở hình dưới là một con chó, ta nhận ra trong hình là con chó thông qua viền của con chó với bức tường đằng sau. Vậy viền là gì.

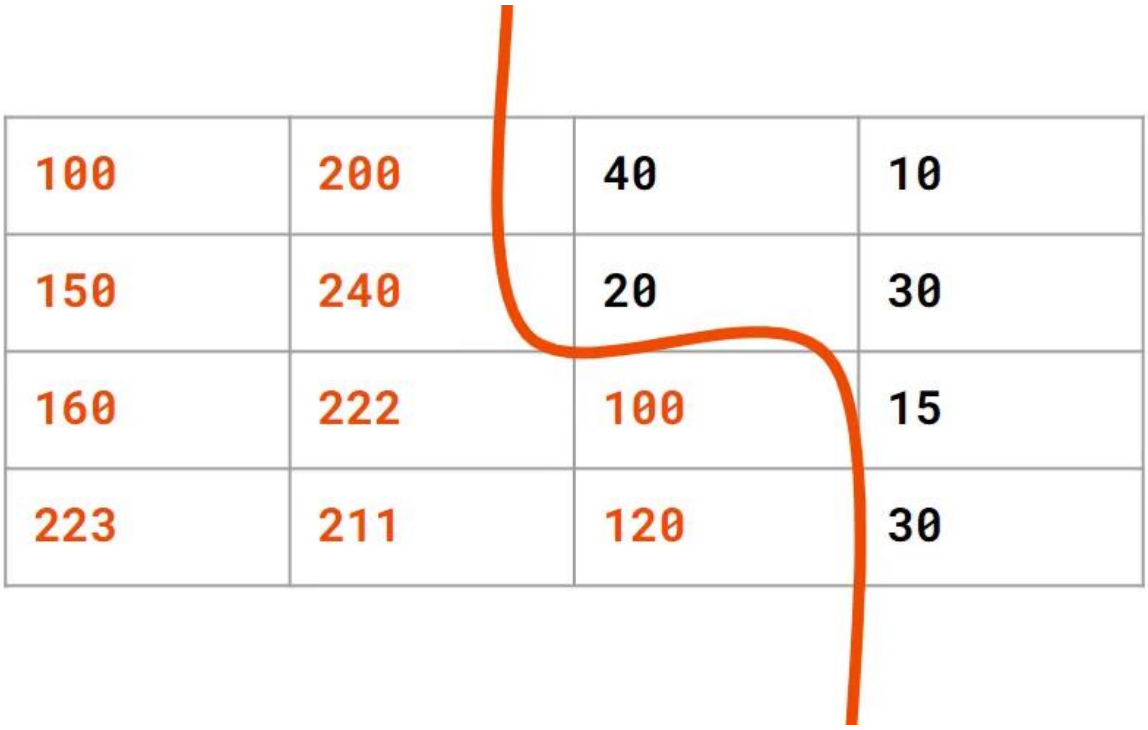
Hình II-9 Phát hiện cạnh-1

Trong xử lý ảnh người ta tập trung rất nhiều vào xử lý viền. Như ở dưới ta chuyển hình ảnh trên thành một đồ thị 3D với 2 chiều là chiều rộng, cao của ảnh và chiều thứ 3 là giá trị pixel thì ta thấy ở mỗi pixel ta phát hiện sự thay đổi giá trị ở các pixel xung quanh, nếu sự thay đổi này lớn. đó chính là cạnh



Hình II-10 Phát hiện cạnh-2

Ở một góc nhìn khác khi ta trải bức ảnh đó ra thì ta thấy rõ ràng hơn



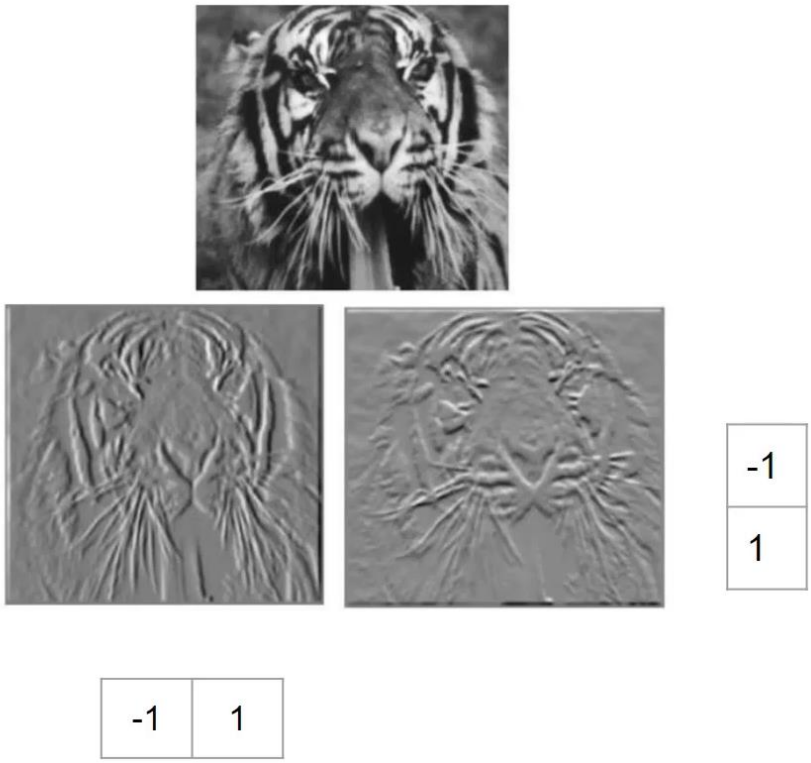
Hình II-11Phát hiện cạnh 3

b. Dùng đạo hàm để phát hiện cạnh – bộ lọc

Ta biết cạnh là một vị trí có sự thay đổi rất nhanh của giá trị pixel, ta cũng biết ảnh có thể coi là một hàm và đạo hàm mô tả sự biến thiên của hàm tại một điểm nào đó. Khi đạo hàm lớn có nghĩa sự thay đổi giá trị của hàm lớn, thật trùng hợp là cạnh cũng vậy. cho nên việc tìm cạnh tương đồng với việc tìm ra đỉnh của đạo hàm.

Để đạo hàm ảnh ta có thể đạo hàm theo biến x hoặc biến y. Để dễ hình dung hơn, ta sử dụng khái niệm bộ lọc để đạo hàm ảnh. dưới là ví dụ cho đạo hàm ảnh. Ta có thể thấy bên trái là đạo hàm theo biến x và bên phải là đạo hàm theo biến y.

Để đạo hàm ta sử dụng một bộ lọc trượt trên ảnh. Ví dụ với đạo hàm theo x, khi ở điểm $(x1, y1)$ ta thực hiện $x1 * -1 + x2 * 1$ rồi ta chuyển sang $(x2, y1)$ cho đến khi hết một dòng ta chuyển lại về đầu của dòng tiếp theo $(x1, y2)$. Phép trượt vừa rồi đó là phép tích chập.



Hình II-12 Đạo hàm của ảnh-1

Ngoài bộ lọc như trên ta có nhiều bộ lọc khác, phổ biến như sobelX và sobelY:

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Hình II-13 Đạo hàm của ảnh-2

c. Cấu trúc mạng CNN

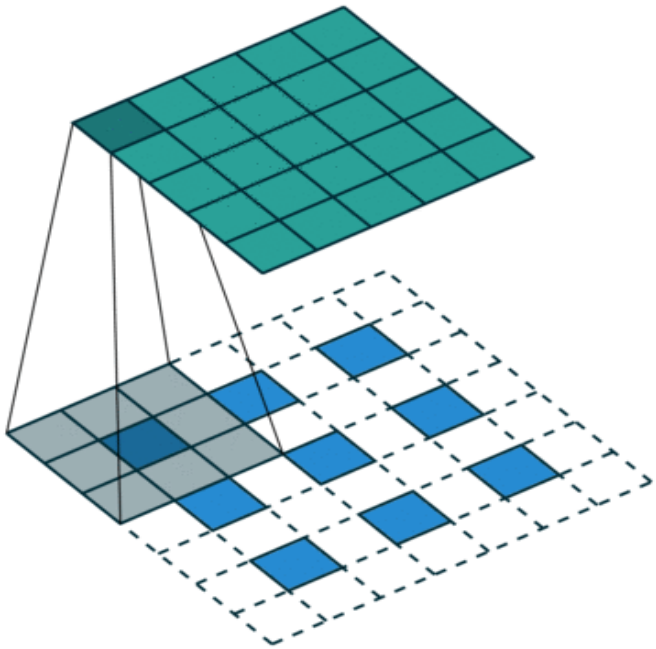
Lớp tích chập gồm có 3 giai đoạn:

- Layer thực hiện song song các biến đổi tuyến tính (tích chập)

- Đầu ra của phép biến đổi tuyến tính này được đưa qua hàm phi tuyến
- Giai đoạn cuối là tổng hợp lại thông tin (Pooling)

- Lớp tích chập

Hình bên dưới là ví dụ cho phép tích chập, bên trên của ảnh là đầu ra của phép tích chập hay còn gọi là feature map, bên dưới có lưới to là ảnh đầu vào, và lưới xám chính là filter.



Hình II-14 Miêu tả cách hoạt động của phép tích chập-1

Khác với khi ta đạo hàm ảnh ta sử dụng một bộ filter cố định thì trong mạng CNN bộ filter sẽ chính là tham số của mô hình, có nghĩa ta sẽ tối ưu bộ filter qua những lần học để tìm được bộ filter tốt nhất.

Có một lưu ý là khi xử lý với ảnh màu. Bộ filter của ta cũng sẽ có 3 kênh màu giống của ảnh và mỗi filter trong bộ là độc lập với nhau.

Ngoài ra khi thực hiện tích chập, để tránh mất thông tin ở viền người ta sử dụng padding. Padding được hiểu là ta thêm một vòng bao bằng 0 bao xung quanh ảnh.

Stride là khoảng cách giữa 2 bước nhảy.

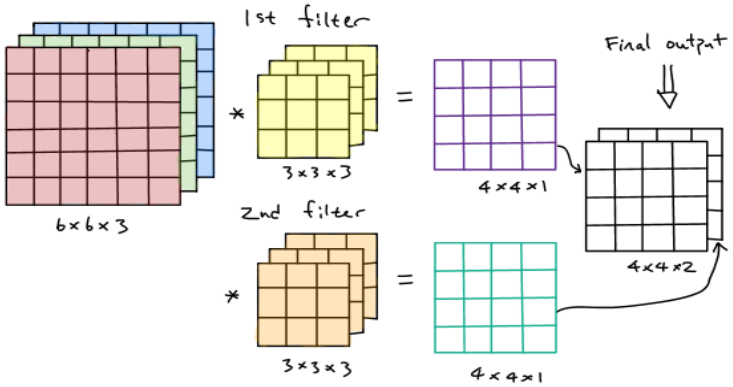
$$Kích\ cỡ\ feature\ map\ = \frac{(W - F + 2P)}{S} + 1$$

W: Chiều dài/rộng của ảnh

F: Kích cỡ bộ lọc

P: Số dòng Padding

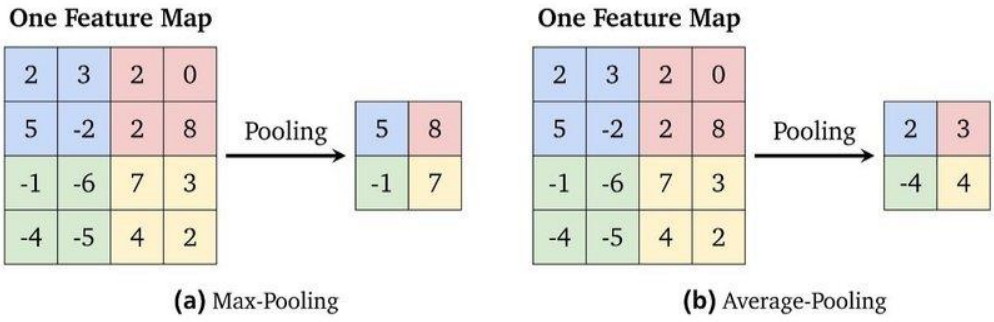
S: Khoảng cách nhảy



Hình II-15 Miêu tả hoạt động của phép tích chập-2

- Lớp Pooling

Pooling có nhiều cách nhưng phổ biến nhất là Max-Pooling và Average-Pooling:



Hình II-16 Phép Pooling

Như hình trên, lớp Pooling sẽ sử dụng các filter và sẽ lấy giá trị lớn nhất hay giá trị trung bình.

Chương III. Xây dựng, huấn luyện mô hình

7. Giới thiệu về tập dữ liệu sử dụng

- Thông tin chung:
    - Tên tập dữ liệu: Face Mask Lite Dataset
    - Tác giả: Prasoon Kottarathil
    - Mô tả: Chứa dữ liệu của người đeo và không đeo khẩu trang
    - Giấy phép: https://creativecommons.org/licenses/by-sa/4.0/
    - Năm phát hành: 2020
    - Gồm 10000 ảnh kích thước 1024x1024 tổng dung lượng 25GB

8. Ngôn ngữ lập trình, thư viện và framework sử dụng

- Ngôn ngữ lập trình: Python

- Môi trường: Google Colab
- Thư viện và framework:
    - Numpy
    - Pygame
    - matplotpib
    - Tensorflow
    - Keras

9. Xây dựng mô hình
    - Đầu tiên, ta sẽ xử lý dữ liệu, chia batch size và chuẩn hoá ảnh:
        - Do kích thước ảnh đầu vào khá lớn 1024x1024 nên ra resize lại ảnh còn 100x100
        - Chia tập huấn luyện – kiểm định theo tỉ lệ 60-40
        - Batch size là 1200
        - Tráo dữ liệu để tổng quát dữ liệu
        - Chuẩn hoá giá trị của ảnh về khoảng 0-1 bằng cách chia cho 255

```
batch_size = 1200
img_height = 100
img_width = 100
train_split = 0.6

train_ds = image_dataset_from_directory(
    data_dir,
    validation_split=1 - train_split,
    subset="training",
    seed=1337,
    image_size=(img_height, img_width),
    batch_size = batch_size,
    shuffle = True
    )
train_ds = train_ds.map(lambda x, y: (Rescaling(1./255)(x), y))

val_ds = image_dataset_from_directory(
    data_dir,
    validation_split=1 - train_split,
    subset="validation",
    seed=1337,
    image_size=(img_height, img_width),
    batch_size = batch_size,
    shuffle = True
    )
val_ds = val_ds.map(lambda x, y: (Rescaling(1./255)(x), y))
```

Hình III-1 Xử lý dữ liệu đầu vào

- Xây dựng mô hình bằng thư viện Tensorflow

Tại bước này em có 2 phiên bản mô hình tuỳ theo mục đích sử dụng

- ở bản đầu tiên ta có mô hình đầu tiên với 10 lớp dung lượng 9.82MB và 2.573.026 tham số

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 100, 100, 16)	448
max_pooling2d_2 (MaxPooling2D)	(None, 50, 50, 16)	0
flatten_2 (Flatten)	(None, 40000)	0
dense_6 (Dense)	(None, 64)	2560064
dropout_4 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 128)	8320
dropout_5 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 32)	4128
dropout_6 (Dropout)	(None, 32)	0
dense_9 (Dense)	(None, 2)	66

Total params: 2573026 (9.82 MB)
Trainable params: 2573026 (9.82 MB)
Non-trainable params: 0 (0.00 Byte)

Hình III-2 Mô hình 1

- Bản thứ 2 ta có mô hình nhẹ hơn chỉ gồm 6 lớp và số node trong mỗi lớp cũng ít hơn

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 6)       168

 max_pooling2d (MaxPooling2  (None, 50, 50, 6)         0
 D)

 flatten (Flatten)           (None, 15000)             0

 dense (Dense)               (None, 43)                645043

 dropout (Dropout)           (None, 43)                0

 dense_1 (Dense)             (None, 2)                 88

=================================================================
Total params: 645299 (2.46 MB)
Trainable params: 645299 (2.46 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Hình III-3 Mô hình 2

- Huấn luyện mô hình
    - Ở mô hình 1 ta huấn luyện với 10 epochs

```
1 model.fit(train_ds, epochs = 10, validation_data =val_ds)
```

```
Epoch 1/10
10/10 [==============================] - 214s 10s/step - loss: 1.5535 - accuracy: 0.5951 - val_loss: 0.3236 - val_accuracy: 0.8752
Epoch 2/10
10/10 [==============================] - 251s 12s/step - loss: 0.2918 - accuracy: 0.9026 - val_loss: 0.1456 - val_accuracy: 0.9955
Epoch 3/10
10/10 [==============================] - 295s 16s/step - loss: 0.1340 - accuracy: 0.9683 - val_loss: 0.0507 - val_accuracy: 0.9989
Epoch 4/10
10/10 [==============================] - 206s 11s/step - loss: 0.0788 - accuracy: 0.9812 - val_loss: 0.0297 - val_accuracy: 0.9992
Epoch 5/10
10/10 [==============================] - 204s 10s/step - loss: 0.0563 - accuracy: 0.9904 - val_loss: 0.0190 - val_accuracy: 0.9996
Epoch 6/10
10/10 [==============================] - 201s 10s/step - loss: 0.0416 - accuracy: 0.9945 - val_loss: 0.0102 - val_accuracy: 0.9998
Epoch 7/10
10/10 [==============================] - 204s 10s/step - loss: 0.0326 - accuracy: 0.9948 - val_loss: 0.0057 - val_accuracy: 0.9999
Epoch 8/10
10/10 [==============================] - 726s 11s/step - loss: 0.0264 - accuracy: 0.9958 - val_loss: 0.0040 - val_accuracy: 0.9999
Epoch 9/10
10/10 [==============================] - 647s 11s/step - loss: 0.0239 - accuracy: 0.9961 - val_loss: 0.0026 - val_accuracy: 0.9999
Epoch 10/10
10/10 [==============================] - 416s 10s/step - loss: 0.0219 - accuracy: 0.9966 - val_loss: 0.0017 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x78642bf849d0>
```

Hình III-4 Huấn luyện mô hình 1

Do tập dữ liệu khá lớn nên rất nhanh độ chính xác trên tập kiểm định đã là 100%

- Ở mô hình 2 do mô hình nặng hơn nên ta chỉ huấn luyện với 5 epochs

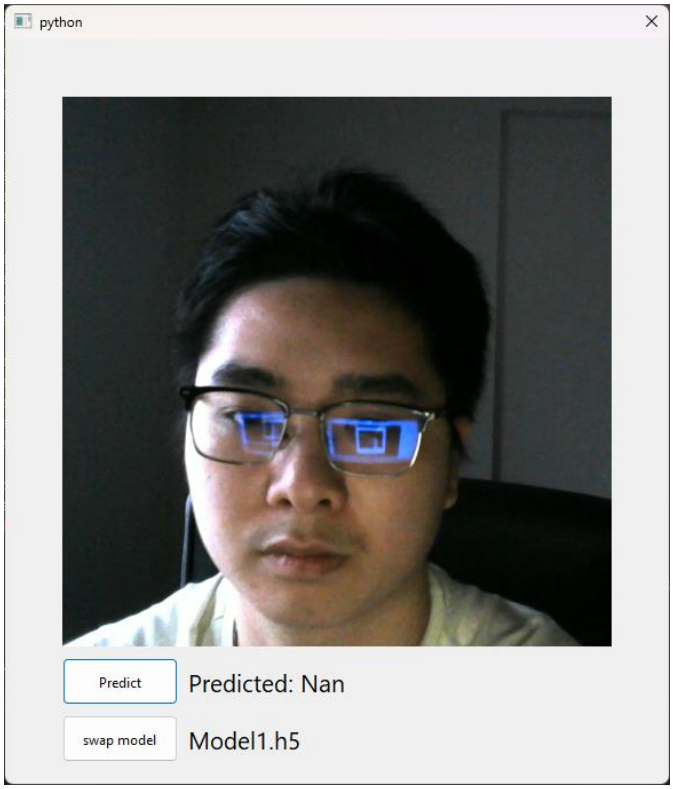
```
model.fit(train_ds, epochs = 5, validation_data=val_ds)
```

```
Epoch 1/5
10/10 [==============================] - 2088s 44s/step - loss: 0.8600 - accuracy: 0.5834 - val_loss: 0.5726 - val_accuracy: 0.9147
Epoch 2/5
10/10 [==============================] - 187s 10s/step - loss: 0.5137 - accuracy: 0.7506 - val_loss: 0.2887 - val_accuracy: 0.9948
Epoch 3/5
10/10 [==============================] - 199s 10s/step - loss: 0.2685 - accuracy: 0.9032 - val_loss: 0.0483 - val_accuracy: 0.9992
Epoch 4/5
10/10 [==============================] - 200s 10s/step - loss: 0.1637 - accuracy: 0.9473 - val_loss: 0.0072 - val_accuracy: 0.9998
Epoch 5/5
10/10 [==============================] - 208s 11s/step - loss: 0.1345 - accuracy: 0.9512 - val_loss: 0.0020 - val_accuracy: 0.9999
<keras.src.callbacks.History at 0x7f2dc321f010>
```

Hình III-5 Huấn luyện mô hình 2

10. Ứng dụng đồ hoạ

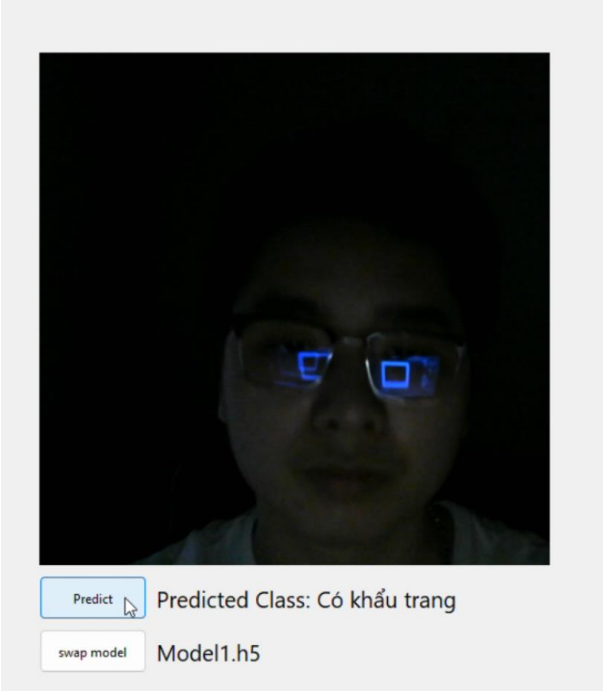
Phần mềm đồ hoạ chúng ta xây dựng từ thư viện PyQt6 của Python. Phần mềm gồm một phần diện tích dùng để chiếu trực tiếp hình ảnh từ camera, một nút "Predict" có nhiệm vụ kích hoạt hàm dự đoán, và một đoạn văn bản hiển thị nhãn là "Không có khẩu trang" và "Có khẩu trang".



Hình III-6 hình ảnh phần mềm đồ hoạ người dùng

11. Đánh giá

Trong file đính kèm gồm có 2 video diễn tả cách hoạt động của ứng đồ hoạ trên từng mô hình đã huấn luyện, ta có thể thấy do mô hình 2 có nhiều lớp hơn giúp phân loại tốt hơn nên dù trong điều kiện thiếu sáng, ứng dụng vẫn nhận biết chính xác. ở mô hình 1 do ít lớp phân loại hơn nên phân loại sẽ kém hơn ở môi trường thiếu sáng. Nhưng trong môi trường đủ sáng, phần mềm vẫn hoạt động hiệu quả.



Hình III-7 Phần mềm với mô hình 1

Hình III-8 Phần mềm với mô hình 2

Chương IV. Kết luận

Kết luận của dự án nghiên cứu về Convolutional Neural Networks (CNN) và ứng dụng của chúng trong việc phát triển một với khả năng phân biệt việc đeo khẩu trang là sự tổng hợp của nỗ lực và kiến thức đã được tích lũy trong quá trình thực hiện dự án.

Trong quá trình nghiên cứu, chúng ta đã tận dụng sức mạnh của CNN, một loại mô hình học sâu được thiết kế đặc biệt để hiểu và rút trích đặc trưng từ dữ liệu hình ảnh. Sự linh hoạt của CNN đã cho phép chúng ta xây dựng một mô hình chính xác và hiệu quả trong việc phân loại xem một khuôn mặt người đó đang đeo khẩu trang hay không.

Tuy nhiên, cũng cần lưu ý rằng còn nhiều thách thức cần vượt qua, như cải thiện độ chính xác trong điều kiện ánh sáng kém, đa dạng về hình dáng khuôn mặt, và bảo mật thông tin cá nhân. Và em kỳ vọng rằng sự tiếp tục nghiên cứu và phát triển sẽ mang lại những cải tiến đáng kể trong tương lai.