ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

ĐỒ ÁN TỔNG HỢP
HƯỚNG TRÍ TUỆ NHÂN TẠO - CO3101

FACE MASK DETECTION

Giảng viên hướng dẫn: Trần Huy
Vũ Văn Tiến

SV thực hiện: Phạm Hoàng Đức Huy – 2011286
Nguyễn Huỳnh Anh Duy – 2011007
Hoàng Ngọc Trí – 2014845
Hoàng Tiến Hải – 2011152

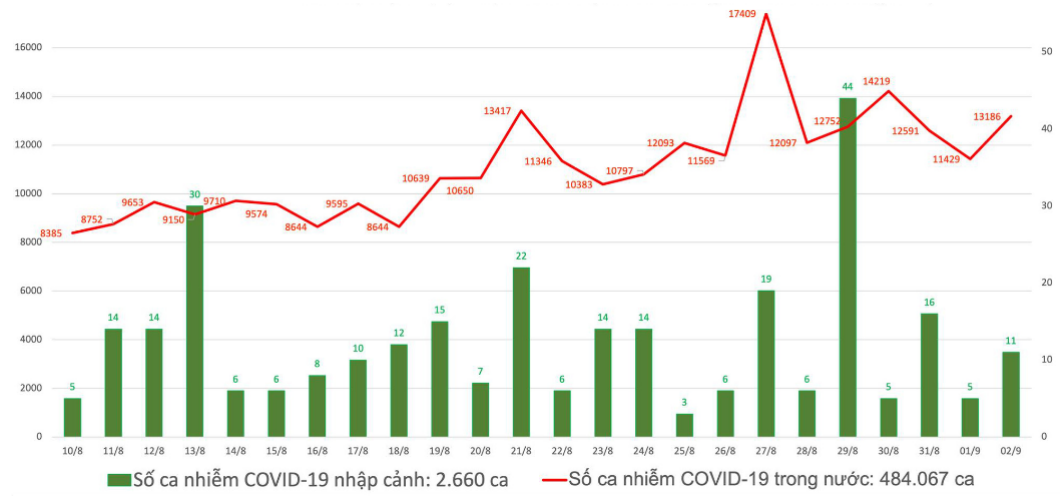
Tp. Hồ Chí Minh, Tháng 12/2022

Mục lục

1 Giới thiệu bài toán

1.1 Hiện trạng/Đặt vấn đề

Trong năm 2021 và nửa đầu năm 2022, dịch bệnh COVID-19 bùng nổ trên toàn cầu với sự lây lan chóng mặt của dịch bệnh này đã cảnh báo cho thế giới một báo động về việc ý thức tự bảo vệ bản thân khỏi nguồn cơn lây nhiễm bằng cách sử dụng khẩu trang để che chắn các bộ phận của bản thân như mũi và miệng giúp hạn chế giọt bắn khi giao tiếp và giảm nguy cơ lây nhiễm virus. Tuy nhiên, sau khi dịch bệnh đã nằm trong tầm kiểm soát và không còn nghiêm trọng như trước, thì việc đeo khẩu trang đã và đang dần trở thành thói quen của nhiều người mỗi khi ở ngoài đường hay ở các khu vực đông đúc.



Hình 1: *Ca nhiễm COVID-19 trong tháng 8-9/2021 tại Việt Nam*

Khẩu trang là một loại mặt nạ bảo vệ được sử dụng để bịt vùng mặt (thường là mũi, miệng) để bảo vệ người đeo khỏi bị lây nhiễm các loại vi khuẩn, dịch bệnh, bụi bặm thông qua đường hô hấp. Khẩu trang y tế được sử dụng nhiều trong các cơ sở y tế, cung cấp cho các bác sĩ, y tá, điều dưỡng viên, giám định pháp y,... đặc biệt là những người làm công việc phẫu thuật, mổ xẻ.

Ngày nay, khẩu trang được sử dụng phổ biến và rộng rãi hơn vì công dụng mà nó mang lại như giúp che nắng, ngăn bụi, giảm được mùi khói xe, khí thải độc hại, hạn chế khói bụi từ ô nhiễm không khí, ngăn chặn mầm bệnh từ người bệnh không phát tán ra bên ngoài và nhằm bảo vệ chính bản thân người mang khẩu trang tránh phải một số bệnh truyền nhiễm qua đường hô hấp từ người khác như các biến thể mới của COVID-19.

Hình 2: *Khẩu trang giúp ngăn chặn virus, bụi bẩn*

Vì vậy ở một số địa điểm nhất định như trên xe bus, bệnh viện hay trong các cơ sở y tế, vẫn có quy định khi bắt buộc người tham gia torng khu vực này phải đeo khẩu trang. Đặc điểm của những khu vực này đều là những nơi đông người, mật độ số người ra vào trong một ngày có thể lên tới hàng trăm, hàng ngàn người. Vì vậy việc kiểm soát mọi người có đeo khẩu trang hay không trở thành một vấn đề khó khăn nếu chỉ sử dụng phương pháp thông thường như có người đứng túc trực ngay cửa ra vào hay nhiều nơi trong khu vực để giám sát và nhắc nhở người khác đeo khẩu trang sẽ không đem lại hiệu quả cao. Một phương pháp mới hơn có thể được áp dụng đó là giám sát tất cả mọi người thông qua camera an ninh được lắp trong từng khu vực và mỗi camera đều được tích hợp phần mềm trích xuất khuôn mặt trong từng khung hình, kiểm tra người xuất hiện trong khung hình đó có đeo khẩu trang hay không và báo cho nhân viên bảo vệ nhắc nhở những trường hợp không đeo khẩu trang.

Hình 3: *Real time Face Mask Detection*

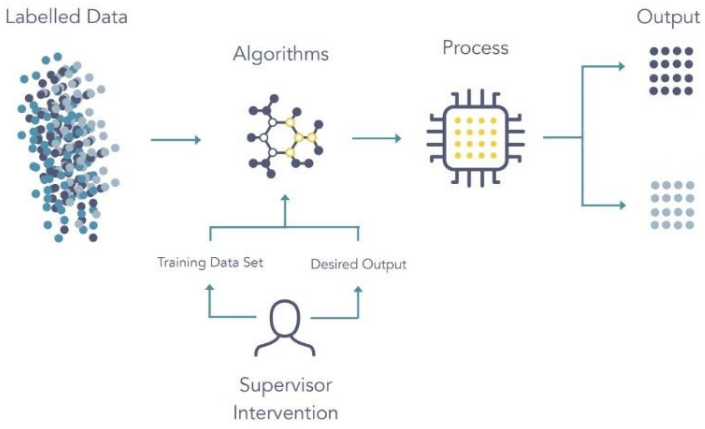
Việc áp dụng những công nghệ hiện nay trong lĩnh vực Deep Learning và phát triển các phần mềm tích hợp tính năng được đề cập ở bên trên là chuyện khả thi để giải quyết vấn đề này sẽ làm cho hiệu quả của công việc được nâng cao và quản lý dễ dàng hơn so với phương pháp thủ công thông thường. Do vậy nhóm sẽ tập trung thực hiện nghiên cứu và hiện thực những mô hình deep learning có khả năng xử lý được hình ảnh mặt người trên từng khung hình và phân loại xem hình ảnh đó có đeo khẩu trang hay không.

1.2 Mục tiêu đề tài

Nhóm sẽ thực hiện đề tài Face Mask Detection với kết quả mong muốn cuối cùng sau khi hoàn thành là xác định người trong khung hình có đeo khẩu trang hay không thông qua camera/webcam theo thời gian thực. Nhóm đã chia nhỏ đề tài thành các bài toán con để có thể hiện thực dễ dàng và trực quan hơn, cụ thể các phần của bài toán như sau:

- Bài toán 1: Trích xuất được vùng có khuôn mặt xuất hiện trong khung hình (đầu vào: hình ảnh, đầu ra: hình ảnh khuôn mặt chính diện).
- Bài toán 2: Nhận diện và phân loại hình ảnh đầu vào có phải là người đang đeo khẩu trang hay không (đầu vào: hình ảnh, đầu ra: nhãn [mask, no mask]).
- Bài toán 3: Tổng hợp kết quả của bài toán 1 và bài toán 2. Thực hiện kết nối giữa hai bài toán trên, sau đó áp dụng vào việc sử dụng mô hình trong thực tế theo thời gian thực trên camera (đầu vào: video, đầu ra: ảnh + nhãn [mask, no mask]).

Trong phạm vi và thời gian của môn học, nhóm sẽ tập trung xử lý bài toán 2 với việc nhận diện hình ảnh được đầu vào sẽ được phân loại rằng người trong ảnh có đeo khẩu trang hay không và gắn nhãn cho ảnh. Bài toán được nhóm giải quyết bằng phương pháp học có giám sát với mạng nơ-ron Convolutional Neural Network và các mô hình học phù hợp cho bài toán này và thu được kết quả như mong muốn.



Hình 4: *Mô hình học có giám sát*

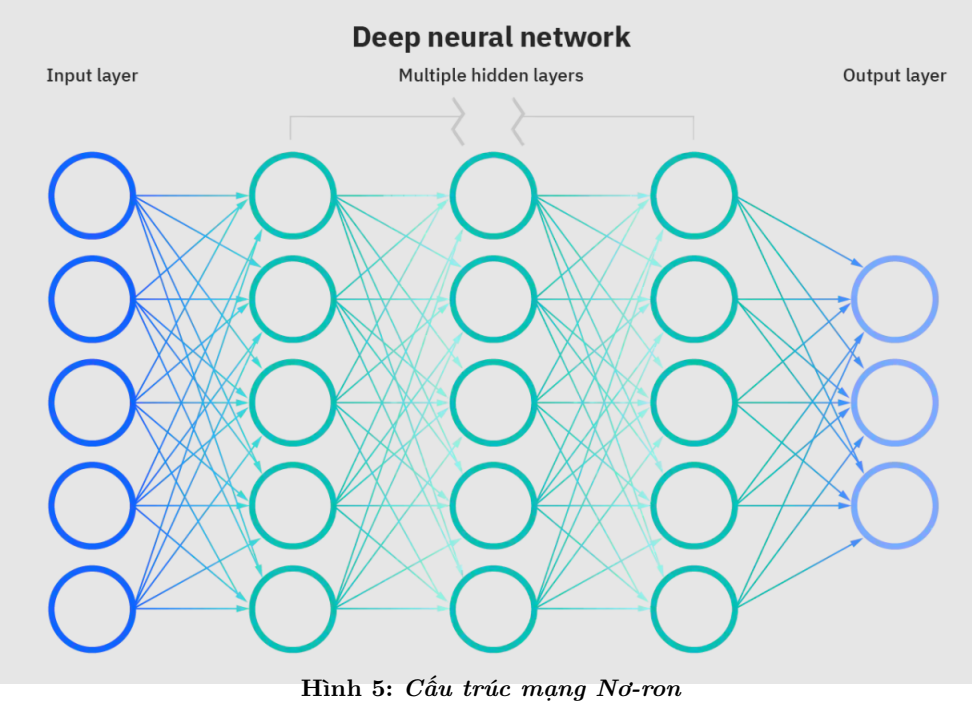
2 Kiến thức tìm hiểu

2.1 Mạng Nơ-ron

Định nghĩa mạng Nơ-ron

Mạng nơ-ron, còn được gọi là mạng nơ-ron nhân tạo (ANN) hoặc mạng nơ-ron mô phỏng (SNN), là một tập hợp con của học máy và là trung tâm của các thuật toán học sâu. Tên và cấu trúc của chúng được lấy cảm hứng từ bộ não con người, bắt chước cách các tế bào thần kinh sinh học truyền tín hiệu cho nhau.

Mạng thần kinh nhân tạo (ANN) bao gồm một lớp nút, chứa lớp đầu vào, một hoặc nhiều lớp ẩn và lớp đầu ra. Mỗi nút, hoặc nơ-ron nhân tạo, kết nối với nút khác và có trọng số và ngưỡng liên quan. Nếu đầu ra của bất kỳ nút riêng lẻ nào cao hơn giá trị ngưỡng đã chỉ định, nút đó sẽ được kích hoạt, gửi dữ liệu đến lớp tiếp theo của mạng. Mặt khác, không có dữ liệu nào được chuyển sang lớp tiếp theo của mạng.



Hình 5: *Cấu trúc mạng Nơ-ron*

Mạng nơ-ron dựa vào dữ liệu đào tạo để học và cải thiện độ chính xác của chúng theo thời gian. Tuy nhiên, một khi các thuật toán học tập này được tinh chỉnh để đạt được độ chính xác, chúng sẽ là những công cụ mạnh mẽ trong khoa học máy tính và trí tuệ nhân tạo, cho phép chúng ta phân loại và phân cụm dữ liệu với tốc độ cao. Các tác vụ trong nhận dạng giọng nói hoặc nhận dạng hình ảnh có thể mất vài phút so với hàng giờ so với nhận dạng thủ công của các chuyên gia con người. Một trong những mạng thần kinh nổi tiếng nhất là thuật toán tìm kiếm của Google.

Cách hoạt động của mạng Nơ-ron

Hãy coi mỗi nút riêng lẻ là mô hình hồi quy tuyến tính của chính nó, bao gồm dữ liệu đầu vào, trọng số, độ lệch (hoặc ngưỡng) và đầu ra. Công thức sẽ giống như thế này:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

$$\text{output} = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

Khi một lớp đầu vào được xác định, các trọng số sẽ được gán. Các trọng số này giúp xác định tầm quan trọng của bất kỳ biến cụ thể nào, với những biến lớn hơn đóng góp đáng kể hơn vào đầu ra so với các đầu vào khác. Tất cả các đầu vào sau đó được nhân với trọng số tương ứng của chúng và sau đó tính tổng. Sau đó, đầu ra được chuyển qua một hàm kích hoạt, xác định đầu ra. Nếu đầu ra đó vượt quá một ngưỡng nhất định, nó sẽ "kích hoạt" nút, chuyển dữ liệu sang lớp tiếp theo trong mạng. Điều này dẫn đến đầu ra của một nút trở thành đầu vào của nút tiếp theo. Quá trình truyền dữ liệu từ lớp này sang lớp tiếp theo xác định mạng thần kinh này là mạng chuyển tiếp (feedforward network).

Khi nghĩ về các trường hợp sử dụng thực tế hơn cho mạng nơ-ron, chẳng hạn như nhận dạng hoặc phân loại hình ảnh, chúng ta sẽ tận dụng phương pháp học có giám sát hoặc bộ dữ liệu được gắn nhãn để đào tạo thuật toán. Khi huấn luyện mô hình, chúng ta sẽ muốn đánh giá độ chính xác của mô hình bằng cách sử dụng hàm chi phí (hoặc tổn thất). Điều này cũng thường được gọi là lỗi bình phương trung bình (MSE).

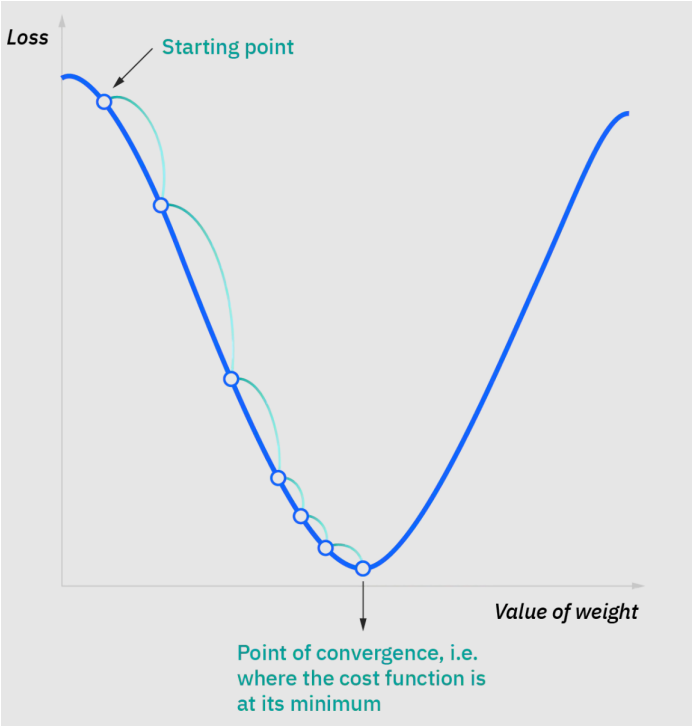
$$Cost\ Function = MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$

Trong phương trình trên:

- i đại diện cho chỉ số của mẫu
- y-hat là kết quả dự đoán

- y là giá trị thực
- m là số lượng mẫu

Cuối cùng, mục tiêu là giảm thiểu hàm chi phí để đảm bảo tính chính xác của sự phù hợp cho bất kỳ quan sát nhất định nào. Khi mô hình điều chỉnh trọng số và độ lệch của nó, nó sử dụng hàm chi phí và học tăng cường để đạt đến điểm hội tụ hoặc mức tối thiểu cục bộ. Quá trình trong đó thuật toán điều chỉnh trọng số của nó là thông qua giảm dần độ dốc, cho phép mô hình xác định hướng cần thực hiện để giảm lỗi (hoặc giảm thiểu hàm chi phí). Với mỗi ví dụ đào tạo, các tham số của mô hình điều chỉnh để dần dần hội tụ ở mức tối thiểu.

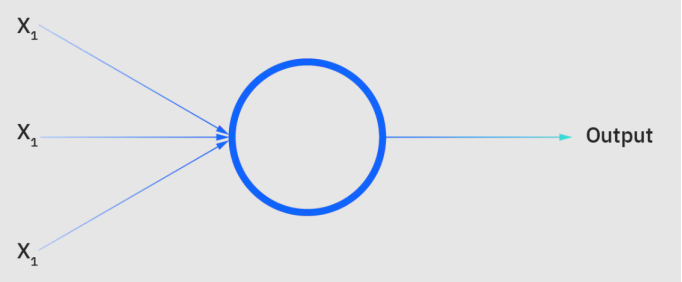


Hầu hết các mạng nơ-ron sâu đều là mạng feedforward, nghĩa là chúng chỉ chảy theo một hướng, từ đầu vào đến đầu ra. Tuy nhiên, chúng ta cũng có thể đào tạo mô hình của mình thông qua lan truyền ngược; nghĩa là di chuyển theo hướng ngược lại từ đầu ra đến đầu vào. Lan truyền ngược cho phép chúng ta tính toán và quy lỗi liên quan đến từng nơ-ron, cho phép chúng ta điều chỉnh và khớp các tham số của (các) mô hình một cách thích hợp.

Một số mạng Nơ-ron tiêu biểu

Perceptron là mạng nơ-ron lâu đời nhất, do Frank Rosenblatt tạo ra vào năm 1958. Nó có một nơ-ron duy nhất và là dạng mạng nơ-ron đơn giản nhất:

Mạng nơ ron chuyển tiếp, hoặc mạng perceptrons nhiều lớp (MLP), bao gồm một lớp



đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra. Mặc dù các mạng nơ-ron này cũng thường được gọi là MLP, nhưng điều quan trọng cần lưu ý là chúng thực sự bao gồm các tế bào nơ-ron sigmoid chứ không phải các perceptron, vì hầu hết các vấn đề trong thế giới thực đều là phi tuyến tính. Dữ liệu thường được đưa vào các mô hình này để đào tạo chúng và chúng là nền tảng cho thị giác máy tính, xử lý ngôn ngữ tự nhiên và các mạng nơ-ron khác.

Mạng nơ-ron tích chập (CNN) tương tự như mạng feedforward, nhưng chúng thường được sử dụng để nhận dạng hình ảnh, nhận dạng mẫu và/hoặc thị giác máy tính. Các mạng này khai thác các nguyên tắc từ đại số tuyến tính, đặc biệt là phép nhân ma trận, để xác định các mẫu trong một hình ảnh.

Mạng nơ-ron tái phát (RNN) được xác định bởi các vòng phản hồi của chúng. Các thuật toán học này chủ yếu được tận dụng khi sử dụng dữ liệu chuỗi thời gian để đưa ra dự đoán về kết quả trong tương lai, chẳng hạn như dự đoán thị trường chứng khoán hoặc dự báo doanh số bán hàng.

2.2 Mạng Nơ-ron tích chập

Định nghĩa mạng Nơ-ron tích chập

CNN là một loại mô hình học sâu để xử lý dữ liệu có dạng lưới, chẳng hạn như hình ảnh, được lấy cảm hứng từ cách tổ chức vỏ não thị giác của động vật và được thiết kế để học tự động và thích ứng các hệ thống phân cấp không gian của các tính năng, từ thấp - đến các mẫu cấp cao.

CNN là một cấu trúc toán học thường bao gồm ba loại lớp (hoặc khối xây dựng): lớp tích chập, lớp tổng hợp và các lớp được kết nối đầy đủ. Hai lớp đầu tiên, lớp tích chập và lớp tổng hợp, thực hiện trích xuất tính năng, trong khi lớp thứ ba, lớp được kết nối đầy đủ, ánh xạ các tính năng được trích xuất thành đầu ra cuối cùng, chẳng hạn như phân loại.

Lớp tích chập đóng một vai trò quan trọng trong CNN, bao gồm một chồng các phép toán, chẳng hạn như tích chập, một loại phép toán tuyến tính chuyên biệt. Trong hình ảnh kỹ thuật số, các giá trị pixel được lưu trữ trong lưới hai chiều (2D), tức là một dãy số và một lưới nhỏ các tham số được gọi là kernel, một trình trích xuất tính năng có thể

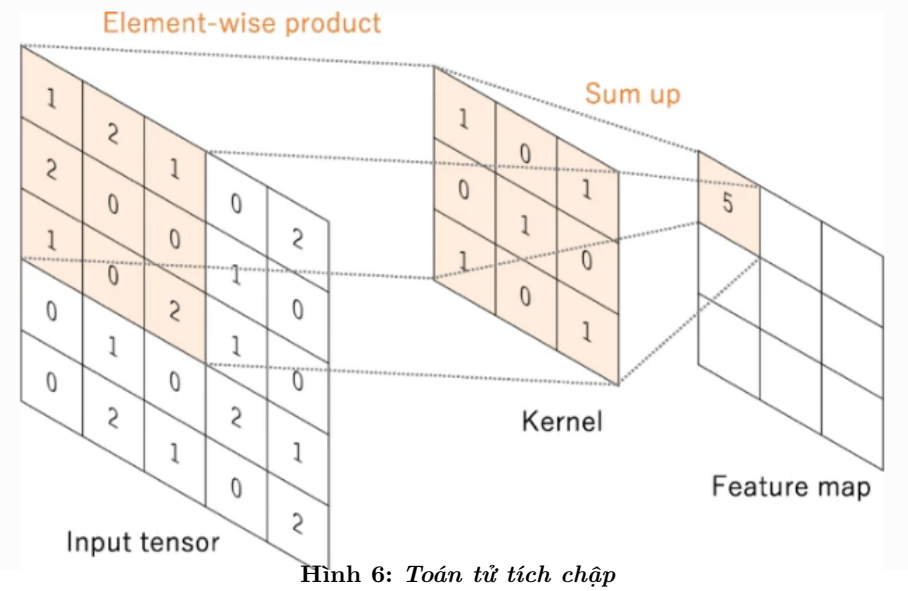
tối ưu hóa, được áp dụng tại mỗi vị trí hình ảnh , làm cho CNN có hiệu quả xử lý hình ảnh cao, vì một đặc điểm có thể xuất hiện ở bất kỳ đâu trong hình ảnh.

Khi một lớp cung cấp đầu ra của nó cho lớp tiếp theo, các tính năng được trích xuất có thể phân cấp và dần dần trở nên phức tạp hơn. Quá trình tối ưu hóa các tham số như hạt nhân được gọi là đào tạo – training, được thực hiện để giảm thiểu sự khác biệt giữa đầu ra và nhãn chân trị cơ bản thông qua một thuật toán tối ưu hóa được gọi là lan truyền ngược và giảm độ dốc, trong số các thuật toán khác.

Các khối trong mạng nơ-ron tích chập

1/ Lớp tích chập

Lớp tích chập là một thành phần cơ bản của kiến trúc CNN thực hiện trích xuất tính năng, thường bao gồm sự kết hợp của các toán tử tuyến tính và phi tuyến tính, tức là toán tử tích chập và hàm kích hoạt.



Hình 6: ***Toán tử tích chập***

Toán tử tích chập

Tích chập là một loại toán tử tuyến tính chuyên biệt được sử dụng để trích xuất tính năng, trong đó một mảng số nhỏ, được gọi là kernel, được áp dụng trên đầu vào, là một mảng số, được gọi là tensor. Một tích phần tử giữa mỗi phần tử của kernel và tensor đầu vào được tính toán tại mỗi vị trí của tensor và tính tổng để thu được giá trị đầu ra ở vị trí tương ứng của tensor đầu ra, được gọi là bản đồ đặc trưng – feature map. Quy trình này được lặp lại khi áp dụng nhiều kernel để tạo thành một số bản đồ đặc trưng tùy ý, biểu thị các đặc điểm khác nhau của các tensor đầu vào, do đó, các hạt nhân khác nhau có thể được coi là các trình trích xuất tính năng khác nhau. Hai siêu tham số chính xác

định hoạt động tích chập là kích thước và số lượng kernal. Kích thước thường là 3 x 3, nhưng đôi khi là 5 x 5 hoặc 7 x 7. Số lượng kernal là tùy ý và xác định độ sâu của bản đồ đặc trưng đầu ra.

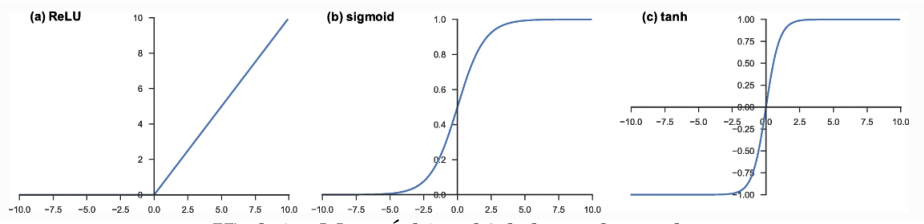
Toán tử tích chập cơ bản không cho phép tâm của mỗi hạt nhân chồng lên phần tử ngoài cùng của tensor đầu vào và làm giảm chiều cao và chiều rộng của bản đồ tính năng đầu ra so với tensor đầu vào. Đệm, thường là đệm bằng 0, là một kỹ thuật để giải quyết vấn đề này, trong đó các hàng và cột chứa các số 0 được thêm vào mỗi bên của tensor đầu vào, sao cho khớp với tâm của nhân trên phần tử ngoài cùng và giữ nguyên trong mặt phẳng kích thước thông qua phép toán tích chập. Các kiến trúc CNN hiện đại thường sử dụng phần đệm bằng 0 để giữ lại các kích thước trong mặt phẳng nhằm áp dụng nhiều lớp hơn. Nếu không có phần đệm bằng 0, mỗi bản đồ tính năng liên tiếp sẽ nhỏ hơn sau thao tác tích chập.

Khoảng cách giữa hai vị trí hạt nhân liên tiếp được gọi là bước tiến - stride, cũng xác định hoạt động tích chập. Sự lựa chọn phổ biến của một stride là 1; tuy nhiên, một stride lớn hơn 1 đôi khi được sử dụng để lấy downsampling các bản đồ đặc trưng. Một kỹ thuật thay thế để thực hiện downsampling xuống là thao tác tổng hợp.

Tóm lại, quá trình đào tạo mô hình CNN liên quan đến lớp tích chập là xác định các hạt nhân hoạt động tốt nhất cho một tác vụ nhất định dựa trên tập dữ liệu đào tạo nhất định. Hạt nhân là tham số duy nhất được học tự động trong quá trình đào tạo trong lớp tích chập; mặt khác, kích thước của hạt nhân, số lượng hạt nhân, phần đệm và bước tiến là các siêu tham số cần được đặt trước khi quá trình đào tạo bắt đầu.

Hàm kích hoạt tuyến tính

Đầu ra của một toán tử tuyến tính chẳng hạn như tích chập sau đó được chuyển qua một chức năng hàm phi tuyến tính. Mặc dù các hàm phi tuyến trơn, chẳng hạn như hàm sigmoid hoặc hyperbolic tangent (tanh), đã được sử dụng trước đây vì chúng là các biểu diễn toán học của hành vi nơ-ron sinh học, nhưng hàm kích hoạt phi tuyến phổ biến nhất được sử dụng hiện nay là đơn vị tuyến tính được chỉnh lưu (ReLU), tính toán đơn giản hàm: f(x) = max(0,x)



Hình 7: ***Một số hàm kích hoạt thông dụng***

2/ Lớp gộp - Pooling Layer

Lớp gộp cung cấp thao tác downsampling điển hình giúp giảm kích thước trong mặt phẳng của các bản đồ đặc trưng nhằm đưa ra đặc điểm bất biến đối với các dịch chuyển và biến dạng nhỏ, đồng thời giảm số lượng tham số có thể học kế tiếp. Cần lưu ý rằng không có tham số có thể học được trong bất kỳ lớp gộp nào, trong khi kích thước bộ lọc, bước tiến và phần đệm là siêu tham số trong hoạt động gộp, tương tự như hoạt động tích chập.

Max Pooling Hình thức hoạt động gộp phổ biến nhất là max pooling, trích xuất các bản vá từ bản đồ tính năng đầu vào, xuất giá trị tối đa trong mỗi patch và loại bỏ tất cả các giá trị khác. Max pooling với bộ lọc có kích thước 2×2 với bước tiến là 2 thường được sử dụng trong thực tế. Thao tác này giảm kích thước trong mặt phẳng của bản đồ đối tượng xuống 2 lần. Không giống như chiều cao và chiều rộng, kích thước độ sâu của bản đồ đối tượng vẫn không thay đổi.

Global average pooling Một hoạt động gộp khác đáng chú ý là gộp trung bình toàn cục. Một phép gộp trung bình toàn cục thực hiện một loại downsampling tuyệt đối, trong đó một bản đồ đối tượng có kích thước chiều cao×chiều rộng được lấy mẫu xuống thành một mảng 1×1 bằng cách lấy trung bình của tất cả các phần tử trong mỗi bản đồ đối tượng, trong khi độ sâu của bản đồ đối tượng là giữ lại. Thao tác này thường chỉ được áp dụng một lần trước khi các lớp được kết nối đầy đủ. Ưu điểm của việc áp dụng gộp trung bình toàn cục như sau: (1) giảm số lượng tham số có thể học được và (2) cho phép CNN chấp nhận đầu vào có kích thước thay đổi.

3/ Lớp kết nối đầy đủ

Các bản đồ đặc trưng đầu ra của lớp tích chập hoặc lớp gộp cuối cùng thường được làm phẳng, tức là, được chuyển đổi thành một mảng một chiều (1D) gồm các số (hoặc vectơ) và được kết nối với một hoặc nhiều lớp được kết nối đầy đủ, còn được gọi là các lớp dày đặc, trong đó mọi đầu vào được kết nối với mọi đầu ra bằng một trọng số có thể học được. Khi các tính năng được trích xuất bởi các lớp tích chập và được lấy mẫu xuống bởi các lớp tổng hợp được tạo, chúng được ánh xạ bởi một tập hợp con các lớp được kết nối đầy đủ với các đầu ra cuối cùng của mạng, chẳng hạn như xác suất cho mỗi lớp trong các lớp phân loại. Lớp kết nối đầy đủ cuối cùng thường có cùng số nút đầu ra với số lớp. Mỗi lớp kết nối đầy đủ được theo sau bởi một hàm phi tuyến tính.

4/ Hàm kích hoạt lớp cuối cùng

Hàm kích hoạt được áp dụng cho lớp kết nối đầy đủ cuối cùng thường khác với các lớp khác. Một hàm kích hoạt thích hợp cần được lựa chọn theo từng task. Các lựa chọn điển hình của lớp cuối cùng chức năng kích hoạt cho các loại tác vụ khác nhau được tóm tắt trong bảng sau.

Task	Last layer activation function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Sigmoid
Regression to continuous values	Identity

2.3 Lan truyền ngược (Backpropagation)

Loss Function Loss Function ký hiệu là L, hay còn gọi là hàm chi phí (hàm lỗi) là hàm số trả về một số thực không âm biểu thị phương pháp đánh giá mức độ hiệu quả của thuật toán khi mô hình hóa tập dữ liệu. Thể hiện sự chênh lệch giữa hai đại lượng $\widehat{y}$ - giá trị mà mô hình dự đoán được, và y - giá trị thực tế. Loss function bắt mô hình phải đóng phạt mỗi lần dự đoán sai và tỉ lệ đóng phạt phụ thuộc vào độ lớn của loss function. Trong mọi bài toán học có giám sát, mô hình phải luôn giảm thiểu giá trị của hàm loss function đến cực tiểu để có được kết quả khả quan nhất thông qua các thuật toán tối ưu và lan truyền ngược.

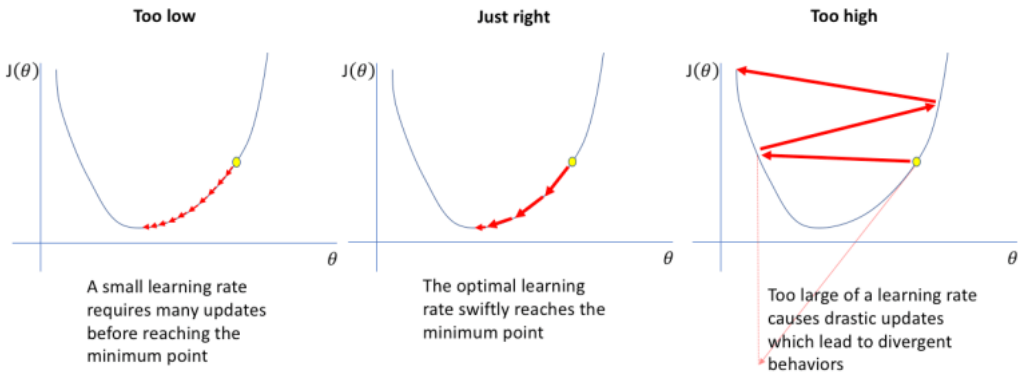
Gradient Descent

Gradient descent là thuật toán tìm giá trị nhỏ nhất của hàm số dựa trên đạo hàm của hàm số đó. Giải thuật này được lặp đi lặp lại để cố gắng tiến dần đến giá trị tối thiểu của hàm số và tìm ra các cực tiểu (cực đại) địa phương. Gradient descent thường được dùng để tối ưu loss function. Thuật toán gradient descent cho hàm số $f(x)$ gồm các bước như sau:

- Bước 1. Khởi tạo giá trị $x = x_0$ tùy ý.
- Bước 2. Gán $x = learning\_rate \cdot f'(x)$ (với learning_rate là một hằng số không âm như 0.001).
- Bước 3. Tính lại hàm số $f(x)$ với giá trị x vừa được cập nhật từ bước 2. Nếu $f(x)$ đủ nhỏ thì dừng thuật toán lại, ngược lại tiếp tục lặp lại bước 2.

Lưu ý khi chọn hệ số learning_rate cũng cực kì quan trọng. Nhưng lưu ý được phân ra thành 3 trường hợp:

- Trường hợp 1: Hệ số learing_rate nhỏ. Mỗi lần hàm số giảm rất ít dẫn đến việc phải lặp lại nhiều lần.
- Trường hợp 2: Hệ số learing_rate phù hợp. Sau một số lần lặp lại thì hàm sẽ đạt giá trị đủ nhỏ.
- Trường hợp 3: Hệ số learing_rate lớn. Gây hiện tượng overshoot và không bao giờ đạt được giá trị nhỏ nhất của hàm.



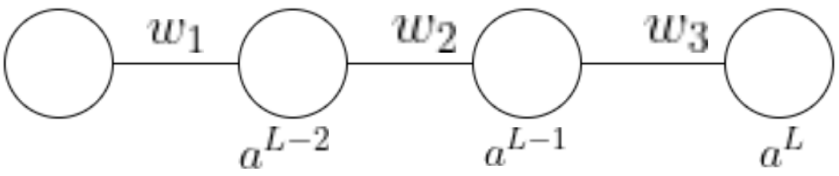
Hình 8: *Các lưu ý khi chọn hệ số Learning_rate*

Thuật toán Backpropagation

Mục tiêu của mô hình neural network là dự đoán chính xác giá trị đầu ra của dữ liệu đầu vào. Để mô hình học được một cách hiệu quả → Cần phải tối ưu hàm mất mát (Loss function). Loss function càng nhỏ, độ tối ưu của mô hình càng cao và kết quả mô hình dự đoán càng chính xác.

Lan truyền ngược là thuật toán dùng để huấn luyện các mô hình neural network một cách hiệu quả thông qua nguyên tắc dây chuyền và thuật toán gradient descent. Nói một cách đơn giản, sau mỗi lần chuyển tiếp qua mạng, lan truyền ngược thực hiện quá trình truyền ngược lại các thông tin được tính toán để điều chỉnh các tham số của mô hình (weights và biases) để tối thiểu hóa loss function trong quá trình huấn luyện mô hình.

Xét một mạng neural đơn giản gồm 2 hidden layer và 1 output layer đều có một node trong mỗi lớp.



Hình 9: *Neural Network đơn giản*

Trong đó:

- $w_i, i \in [1, 3]$: Trọng số (weight) của mỗi node truyền qua node tiếp theo.
- $a^{L-i}, i \in [0, 2]$: Giá trị của mỗi node khi đi qua hàm activation (sigmoid, relu, ...) tính từ output layer trở về input layer.

Hàm loss function được sử dụng trong mô hình neural network này là hàm trung bình

bình phương có dạng như sau:

$$L = \frac{1}{2}(\widehat{y} - y)^2$$

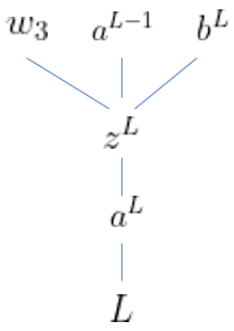
Với:

- $\widehat{y}$: Giá trị output dự đoán của mô hình.
- y: Giá trị thực tế của dữ liệu.

Giá trị của hàm loss khi đi qua mạng neural sẽ được tính theo công thức được đề cập ở trên với các tham số từ lớp output layer.

$$L = \frac{1}{2}(a^L - y)^2 \ = \frac{1}{2}(\sigma(z^L) - y)^2$$

Trong đó z^L là gía trị được tính toán dựa trên weight và bias từ lớp phía trước đó, giá trị này chưa được đưa qua hàm activation: $z^L = w_3 \cdot a^{L-1} + b^L$.



Hình 10: *Sơ đồ liên hệ giữa các tham số và output A*

Để tối thiểu hóa loss function bằng các sử dụng thuật toán gradient descent thường được sử dụng trong các mô hình học máy giúp việc tính toán trở nên trực quan. Tiếp theo tiến hành tìm đạo hàm của hàm loss dựa trên quy tắc chain rule để tính đạo hàm riêng phần theo từng tham số. Từ đó có thể thấy được độ nhạy của hàm loss khi các giá trị tham số thay đổi như thế nào.

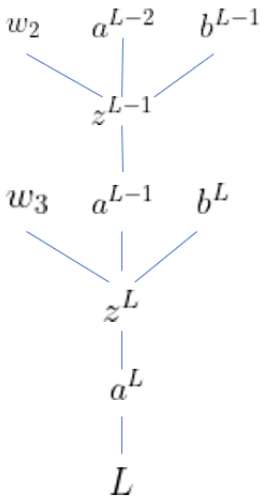
$$\frac{\partial L}{\partial w_3} = \frac{\partial z^L}{\partial w_3} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial L}{\partial a^L} = a^{L-1} \cdot \sigma'(z^L) \cdot (a^L - y)$$

$$\frac{\partial L}{\partial b^L} = \frac{\partial z^L}{\partial b^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial L}{\partial a^L} = 1 \cdot \sigma'(z^L) \cdot (a^L - y)$$

$$\frac{\partial L}{\partial a^{L-1}} = \frac{\partial z^L}{\partial a^{L-1}} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial L}{\partial a^L} = w_3 \cdot \sigma'(z^L) \cdot (a^L - y)$$

Theo công thức đạo hàm riêng phần của hàm loss theo từng tham số, có thể thấy được mọi tham số đều có thể ảnh hưởng tới hàm loss. Ảnh hưởng nhiều nhất đến độ nhảy của

hàm loss là hai giá trị w_3 (trọng số để tính giá trị z^L cho node output) và a^{L-1} (giá trị của node phía trước). Ta có thể điều chỉnh trọng số w_3 và b^L một cách dễ dàng khi đây là các số thực có thể chỉnh sửa trực tiếp, nhưng giá trị a^{L-1} là giá trị của node trước đó nên không thể trực tiếp thay đổi mà phải thông qua điều chỉnh w_2 và b^{L-1}. Từ đó ý tưởng lan truyền ngược xuất hiện để điều chỉnh trọng số của các node trước đó trong quá trình tối thiểu hóa loss function.



Hình 11: *Sơ đồ liên hệ giữa các tham số và output B*

Công thức điều chỉnh trọng số (weight) và bias mỗi lần lan truyền ngược:

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w_{old}}$$

$$b_{new} = b_{old} - \eta \cdot \frac{\partial L}{\partial b_{old}}$$

Trong đó: η là learning rate.

Quá trình lan truyền ngược sẽ liên tục lặp đi lặp lại qu1 trình điều chỉnh các weights và biases trong mạng neural để tìm giá trị nhỏ nhất của loss function với mục đích tối thiểu hóa sự sai lệch giữa giá trị dự đoán của mô hình và giá trị thực tế của output.

3 Mạng Lenet

3.1 Sơ lược về mạng Lenet

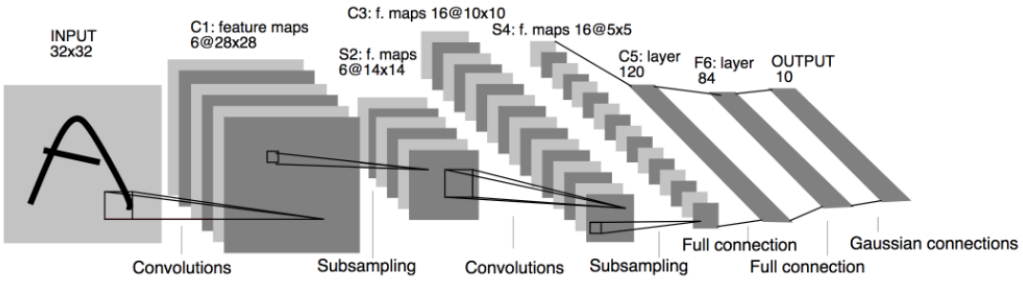
LeNet là một cấu trúc mạng thần kinh tích chập được đề xuất bởi LeCun et al năm 1998, đây là một trong những mạng thần kinh tích chập sớm nhất và đã thúc đẩy sự phá triển của deep learning.

Từ năm 1988, sau nhiều năm nghiên cứu và nhiều lần lặp lại thành công, công trình tiên phong được đặt tên là LeNet-5.

Đến năm 1998, khi đã xem xét các phương pháp khác nhau được áp dụng để nhận dạng ký tự viết tay và so sánh chúng với các tiêu chuẩn nhận dạng chữ số viết tay tiêu chuẩn. Kết quả cho thấy mạng vượt trội hơn tất cả các mô hình khác.

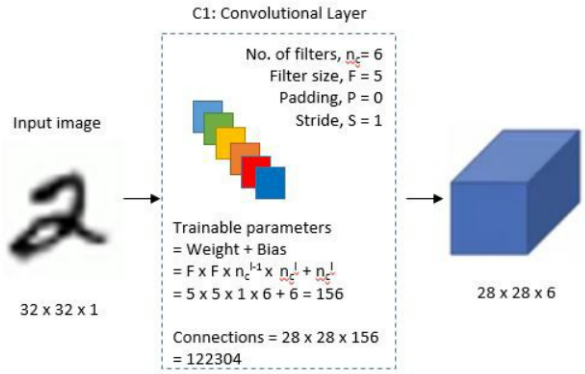
3.2 Kiến trúc mạng Lenet-5

Kiến trúc LeNet-5 bao gồm 7 lớp trong đó: có hai bộ lớp tích chập và gộp trung bình, tiếp theo là lớp tích chập làm phẳng, sau đó là một lớp kết nối đầy đủ và cuối cùng là một lớp đầu ra softmax được kết nối đầy đủ.



Hình 12: *Mạng Lenet-5 [LeCun et al., 1998]*

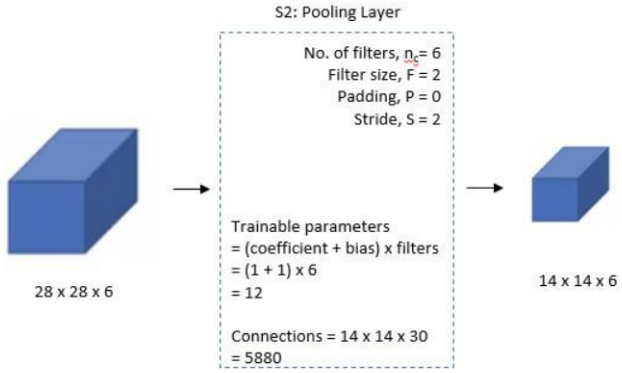
Ảnh đầu vào là 1 bức ảnh thang độ xám kích thước 32x32 pixel. Ở tầng chập C1 chúng ta sử dụng 6 ma trận chập kích thước 5x5 cho ra 6 ma trận ảnh đặc trưng sau khi chập lần 1 đó là các ma trận ánh xạ đặc trưng ở tầng chập C1, mỗi ma trận ánh xạ đặc trong này có kích thước 28x28. Tức là ảnh gốc ban đầu được phân tích theo 6 chiều đặc trưng khác nhau với ma trận chập 5x5.



Hình 13: *Lớp tích chập đầu tiên (C1)*

Do kích thước các ảnh đặc trưng ở tầng chập C1 có kích thước 28x28 còn lớn, cho nên bước tiếp theo chúng ta thực hiện phép giảm số chiều ở ma trận đặc trưng (Pooling) với hệ số tỷ lệ là 2 sử dụng hàm max.

Như vậy với 6 ma trận đặc trưng kích thước 28x28 ở tầng chập C1 ta tạo được 6 ma trận kích thước 14x14 ở tầng subsampling (S2).



Hình 14: *Lớp gộp trung bình thứ nhất (S2)*

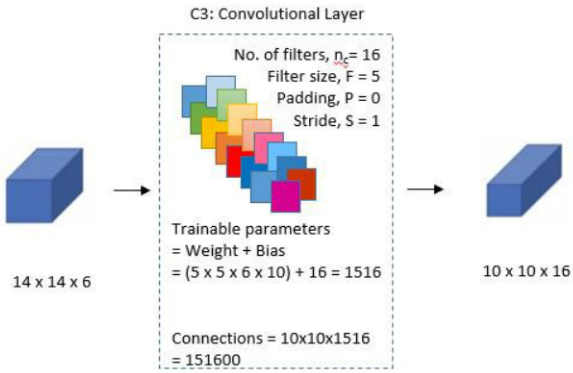
Tiếp tục sử dụng 60 ma trận chập kích thước 5x5 chập với các ma trận ở tầng S2 ta được 16 ma trận ánh xạ đặc trưng kích thước 10x10 ở tầng chập C3. Trong lớp này, các ma trận trong 16 bản đồ đặc trưng được kết nối với 6 bản đồ đặc trưng của lớp trước như sau:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

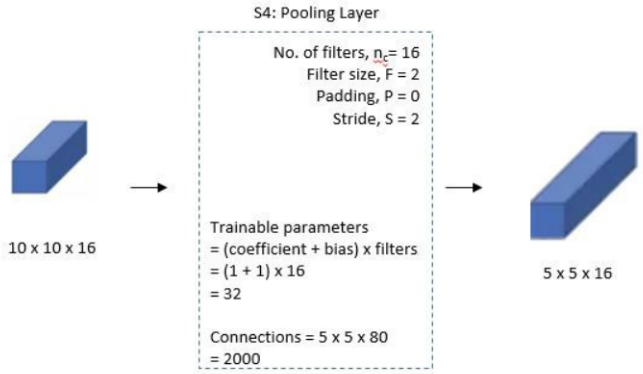
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Lý do cho việc phân chia như vậy chính là để phá vỡ tính đối xứng trong mạng và giữ số lượng kết nối trong giới hạn hợp lý. Đó là lý do tại sao số lượng tham số đào tạo trong các lớp này là 1516 thay vì 2400 và tương tự, số lượng kết nối là 151600 thay vì 240000.



Hình 15: *Lớp tích chập thứ hai (C3)*

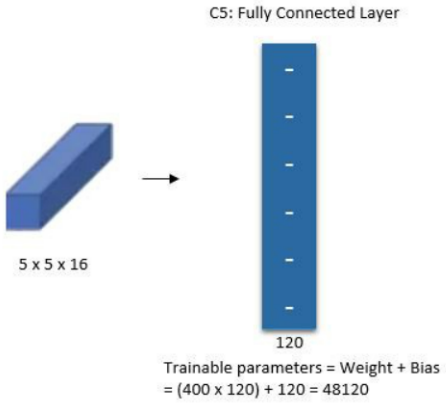
Do kích thước các ảnh đặc trưng ở tầng chập C3 có kích thước 10x10 còn lớn, cho nên bước tiếp theo chúng ta thực hiện phép giảm số chiều ở ma trận đặc trưng (Pooling) với hệ số tỷ lệ là 2 sử dụng hàm max.



Hình 16: *Lớp gộp trung bình thứ hai (S4)*

Kết quả với 16 ma trận đặc trưng kích thước 10x10 ở tầng chập C3 ta tạo được 16 ma trận kích thước 5x5 ở tầng subsampling (S4)

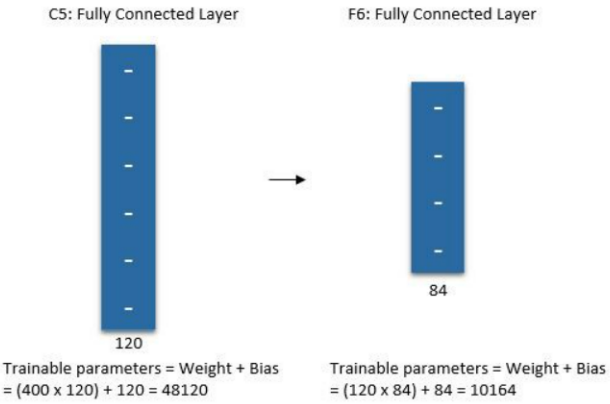
Tiếp tục sử dụng 16 X 120 = 1920 ma trận chập kích thước 5x5 chập với các ma trận ở tầng S4 ta được 120 ma trận ánh xạ đặc trưng kích thước 1x1 ở tầng chập C5. Trong lớp này, mỗi ma trận trong 120 bản đồ đặc trưng được kết nối với tất cả 16 bản đồ đặc trưng của lớp trước.



Hình 17: *Lớp tích chập làm phẳng (C5)*

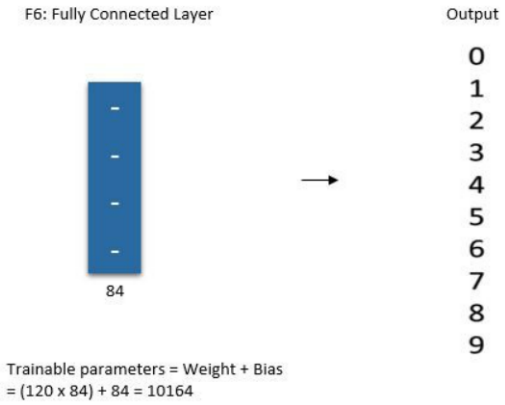
Do các đặc trưng ở tầng chập C5 là các điểm đặc trưng 1x1, cho nên ta không thực hiện phép toán subsampling nữa.

Tiếp theo ta sử dụng phép toán max để giảm kích thước ở tầng chập C5 do tầng C5 có tới 120 node đặc trưng, ta dùng hàm max giảm xuống còn 84 node ở tầng F6.



Hình 18: *Lớp kết nối đầy đủ (F6)*

Cuối cùng, có một lớp đầu ra softmax được kết nối đầy đủ ỹ với 10 giá trị có thể tương ứng với các chữ số từ 0 đến 9.



Hình 19: *Lớp đầu ra (OUTPUT)*

Tuy nhiên do tính chất bài toán, khi áp dụng mạng nhóm chỉ triển khai 2 ouput là : WithMask và WithoutMask.

4 Thiết kế và hiện thực

4.1 Môi trường lập trình và công cụ hỗ trợ

- Môi trường lập trình: Google Codelabs
- Ngôn ngữ sử dụng: Python
- Thư viện hỗ trợ: Tensorflow, keras, cv2, numpy, ...

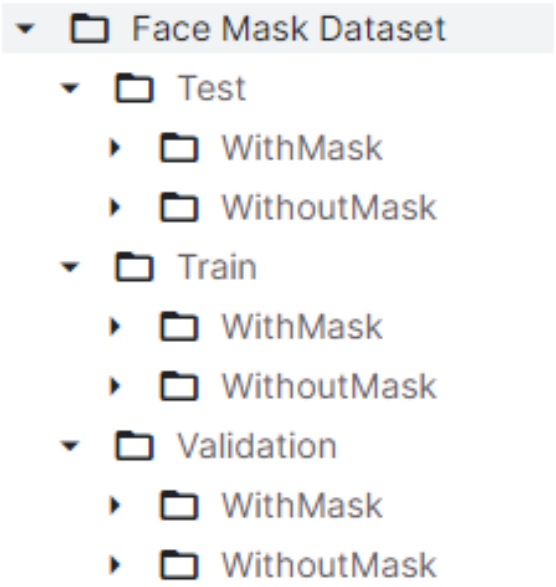
4.2 Mô tả dữ liệu

Cơ sở dữ liệu hình ảnh thu thập tại:

- Face Mask Detection 12K Images Dataset (https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset)

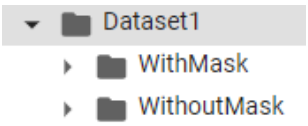
Cấu trúc tập dữ liệu:

- Tập dữ liệu chứa tổng cộng 11792 hình ảnh khuôn mặt đeo khẩu trang và khuôn mặt không đeo khẩu trang, hình ảnh màu, chụp cận mặt, các hình ảnh có các kích thước khác nhau. Trong đó, 5883 hình ảnh khuôn mặt đeo khẩu trang và 5909 hình ảnh khuôn mặt không đeo khẩu trang.
- Dữ liệu được chứa trong 3 thư mục là Test, Train và Validation. Mỗi thư mục đều bao gồm 2 thư mục là:
  - WithMask: chứa hình ảnh khuôn mặt đeo khẩu trang.
  - WithoutMask: chứa hình ảnh khuôn mặt không đeo khẩu trang.



- Thư mục Test có 483 hình ảnh trong thư mục WithMask và 509 hình ảnh trong thư mục WithoutMask, thư mục Train có 5000 hình ảnh trong mỗi thư mục và thư mục Validation có 400 hình ảnh trong mỗi thư mục.

Với mong muốn tự thực hiện việc phân chia các tập dữ liệu Train, Validation và Test, nhóm đã thực hiện cấu trúc lại tập dữ liệu. Toàn bộ dữ liệu 11792 hình ảnh được chứa trong 2 thư mục là WithMask và WithoutMask. Trong đó, thư mục WithMask có 5883 hình ảnh khuôn mặt đeo khẩu trang và thư mục WithoutMask có 5909 hình ảnh khuôn mặt không đeo khẩu trang.



4.3 Tiền xử lý dữ liệu

Các bước tiền xử lý dữ liệu bao gồm:

- Sau khi import các thư viện, thực hiện khai báo các tham số để tiến hành load dữ liệu và định dạng kích thước ảnh.

```
DIRECTORY = r"/content/drive/MyDrive/Do_an_AI/Dataset1"
CATEGORIES = ["WithMask", "WithoutMask"]
data = []

img_size = 32
BATCH_SIZE = 32;
```

- Trong quá trình load dữ liệu, biến đổi ảnh màu thành ảnh xám (grayscale) để phù hợp với mô hình Lenet-5.

```
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    class_num = CATEGORIES.index(category)
    for img in  os.listdir (path):
        img_path = os.path.join(path, img)
        image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        image_array = cv2.resize(image,(img_size, img_size))

        data.append([image_array,class_num])
```

- Đưa tất cả ảnh về cùng kích thước 32x32, thêm vào 1 chiều cho dữ liệu để thể hiện color channel gray. Chuyển đổi ảnh và nhãn sang định dạng mảng để thuận tiện, nhanh chóng trong quá trình làm việc về sau. Data chưa được normalized nên ta cần chia mọi pixels của ảnh cho 255 để đưa dữ liệu về khoảng [0,1], đây là một bước xử lý cơ bản cho ảnh.

```
images = []
labels  = []
for image, label in data:
    images.append(image)
    labels.append(label)

images = np.array(images, dtype="float32").reshape(-1, img_size, img_size,
1)
images = images / 255.0;
```

```
labels = np.array(labels)
```

- Format lại tập label theo dạng a one-hot vector để dùng cho quá trình training.

```
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
```

- Chia tập dữ liệu ra thành 3 tập: train, validation và test.

```
(trainX, testX, trainY, testY) = train_test_split(images, labels, test_size
=0.20, stratify =labels, random_state=10)
(trainX, valX, trainY, valY) = train_test_split(trainX, trainY, test_size=0
.25, random_state=10)
```

  - Đầu tiên, chia dữ liệu 80% tập Train và 20% tập Test
  - Tiếp theo, chia tập Train thành 75% tập Train và 25% tập Validation.
  - Kích thước các tập sau khi chia:

```
trainX.shape
```

```
(7074, 32, 32, 1)
```

```
trainY.shape
```

```
(7074, 2)
```

```
valX.shape
```

```
(2359, 32, 32, 1)
```

```
testX.shape
```

```
(2359, 32, 32, 1)
```

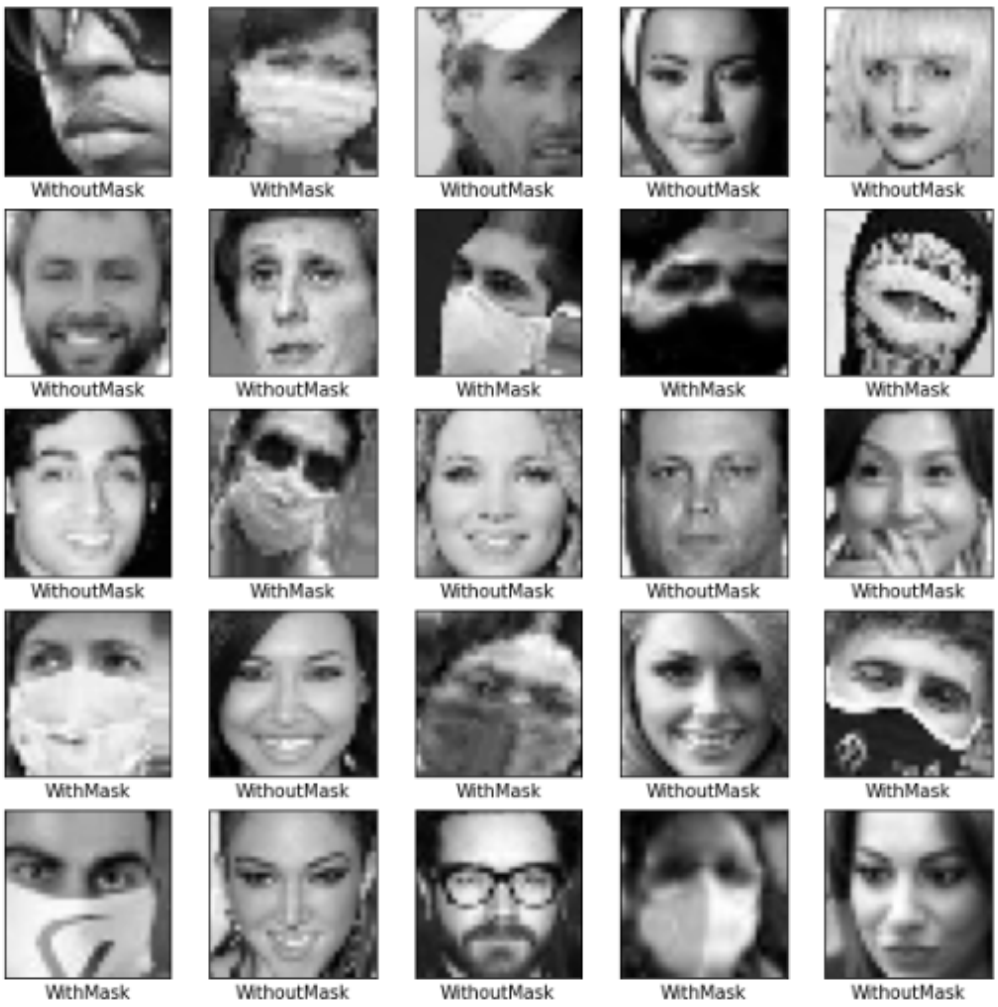
- Tạo dictionary để giải mã label thành nhãn tương ứng

```
encode = {0: 'WithMask', 1: 'WithoutMask'}

```

Vẽ 25 hình đầu tiên trong training set với label tương ứng để kiểm tra trước khi xây dựng mô hình thực hiện training.

```
plt.figure ( figsize =(10,10))
for i in range(25):
    image = trainX[i].reshape(img_size, img_size)
    plt.subplot(5 ,5,i +1)
    plt.xticks ([])
```

```
        plt.yticks ([])
        plt.grid (False)
        plt.imshow(image , cmap='gray')
        plt.xlabel (encode[int(trainY[i ][[1]]) ])

    plt.show()
```



4.4 Xây dựng mô hình

```
model = tf.keras.Sequential()

model.add(tf.keras.layers.Conv2D(6, kernel_size=(5, 5), activation='relu',
input_shape=(img_size, img_size, 1)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Conv2D(16, kernel_size=(5, 5), activation='relu'))
```

```
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
    model.add( tf.keras.layers.Flatten ())
    model.add(tf.keras.layers.Dense(120, activation='relu'))
    model.add(tf.keras.layers.Dense(84, activation='relu'))
    model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

Sử dụng hàm **summary** để có cái nhìn tổng thể về model

```
    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         156

 max_pooling2d (MaxPooling2D  (None, 14, 14, 6)        0
 )

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 16)         0
 2D)

 flatten (Flatten)           (None, 400)               0

 dense (Dense)               (None, 120)               48120

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 2)                 170

=================================================================
Total params: 61,026
Trainable params: 61,026
Non-trainable params: 0
_________________________________________________________________
```

Định nghĩa các Layer trong mô hình và các thành phần được sử dụng trong mạng Lenet-5:

- Lớp Conv2D đầu tiên: Đây là input layer của mạng, là Conv2D filters có kernel (5x5) nhận input có kích thước 32 x 32, là một ảnh size 32 x 32 và chỉ có 1 màu (grayscale). Layer giữ nguyên kích thước ban đầu của ảnh bằng padding, và tạo ra 6 convoluted images cùng kích thước với ảnh ban đầu.
- Lớp MaxPooling2D: Các convoluted images được thu nhỏ lại với kích thước giảm một nửa bằng max pooling với stride = 2.
- Lớp Flatten: Đây là layer để "làm phẳng" hình ảnh trước khi đưa vào Dense layer. Layer này chuyển ảnh từ mảng 3 chiều thành mảng 1 chiều kích thước 1024 pixels (= 32 x 32). Layer này không có tham số để học mà chỉ định dạng lại dữ liệu.
- Lớp Dense đầu tiên: Đây là fully connected layer có 120 neurons, lấy input từ layer

trước của nó. Mỗi neuron lấy input từ tất cả output layer trước, điều chỉnh lại trọng số cần học và output ra 1 giá trị vào layer kế tiếp.

- Lớp Dense cuối cùng (output): Đây là fully connected layer với 2 node, mỗi node tương ứng với 1 label là mask hoặc no mask. Layer này có input từ 84 nodes của layer ngay phía trước và output một giá trị trong khoảng [0..1]. Tổng của 2 node này cho ra giá trị có kết quả bằng 1.

Tiếp theo, nhóm định nghĩa callback. Callback này được truyền vào tham số callback của mô hình huấn luyện, với mục đích dừng sớm quá trình huấn luyện khi giá trị validation loss không thay đổi tốt hơn (giữ nguyên hoặc tăng lên) khi chạy được tối đa 4 epoch.

```
callback = tf.keras.callbacks.EarlyStopping(
    monitor = "val_loss",
    min_delta = 0,
    mode = "auto",
    patience = 4,
    restore_best_weights = False,
)
```

Việc cuối cùng đó là tiến hành compile model để biên dịch mô hình trước khi đưa vào quá trình huấn luyện:

```
lr=2e-4
model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adam(lr),
metrics=['accuracy'])
```

- Loss function: Thuật toán dùng để đo đạc xem output của model sai khác bao nhiêu so với output mong muốn. Mục tiêu của training là giảm thiểu hàm loss function. Binary_crossentropy trong trường hợp cơ bản chỉ có hai lớp sẽ mặc định giá trị đúng của hai lớp đó là 0 hoặc 1. Sau đó hàm sẽ thực hiện tính mức phạt dựa trên sự chênh lệch giữa xác suất dự đoán và xác suất đúng.
- Optimizer: Thuật toán điều chỉnh tham số bên trong của model để giảm thiểu hàm loss function. Adam là optimizer phổ biến cho các model đơn giản với learning rate là $2 \cdot 10^{-4}$
- Metrics: Dùng để giám sát quá trình training và là thước đo để đánh giá hiệu suất của một mô hình. accuracy là tỉ lệ hình ảnh được gắn nhãn đúng.

Huấn luyện model: Nhóm sử dụng hàm fit() để huấn luyện mô hình Lenet-5 vừa được xây dựng ở trên.

```
history= model.fit(trainX, trainY, batch_size=BATCH_SIZE,
```

```
    epochs=50, verbose=1, callbacks=[callback], validation_data=(valX, valY))
```

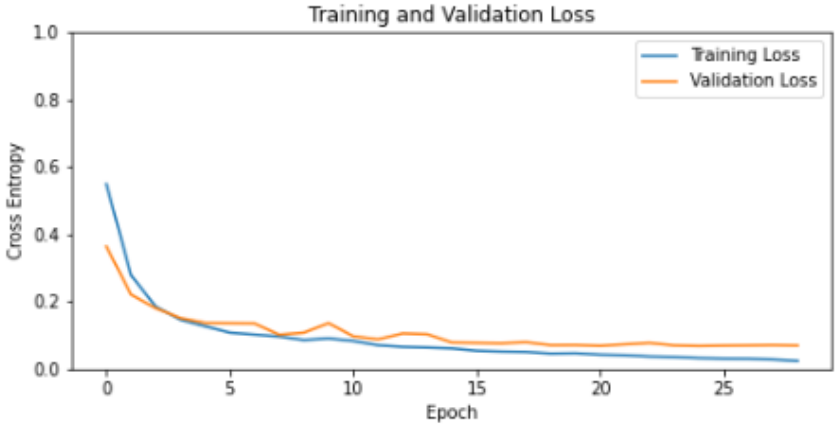
4.5 Kết quả

Kết quả và thời gian huấn luyện:

```
Epoch 26/50
222/222 [==============================] - 6s 29ms/step - loss: 0.0307 - accuracy: 0.9890 - val_loss: 0.0691 - val_accuracy: 0.9763
Epoch 27/50
222/222 [==============================] - 6s 29ms/step - loss: 0.0303 - accuracy: 0.9888 - val_loss: 0.0698 - val_accuracy: 0.9771
Epoch 28/50
222/222 [==============================] - 7s 32ms/step - loss: 0.0277 - accuracy: 0.9904 - val_loss: 0.0706 - val_accuracy: 0.9767
Epoch 29/50
222/222 [==============================] - 8s 35ms/step - loss: 0.0235 - accuracy: 0.9921 - val_loss: 0.0694 - val_accuracy: 0.9784
184.93365597724915
```

Kết quả dừng lại ở EPOCH số 29 do nhận thấy giá trị loss của validation không cải thiện sau 4 epoch.

Trực quan kết quả huấn luyện bằng đồ thị:



Đường màu xanh đại diện cho tập Train, còn màu cam đại diện cho tập Validation. Nhìn chung kết quả huấn luyện càng tốt hơn qua mỗi EPOCH với độ chính xác lớn.

In ra 30 ảnh trên tập Test và so sánh kết quả dự đoán, dự đoán đúng nhãn có màu xanh và dự đoán sai nhãn có màu đỏ. Các ảnh đều được dự đoán chính xác có đeo khẩu trang hay không.

Độ chính xác trên tập Test đạt được:

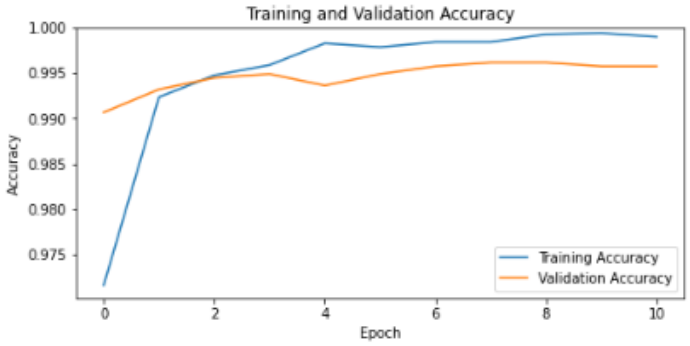
```
74/74 [==============================] - 1s 12ms/step - loss: 0.0819 - accuracy: 0.9729
Model accuracy: 97.29%
```

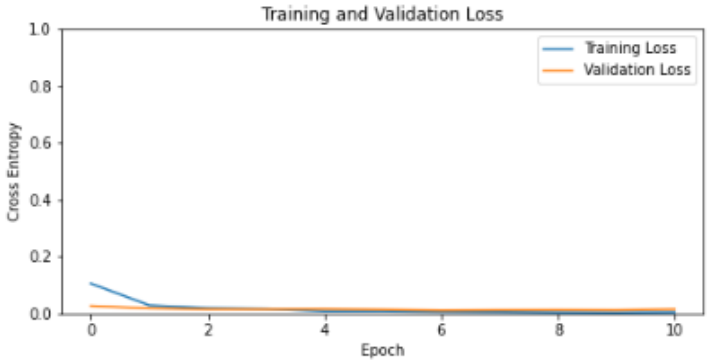
Có thể thấy model chúng ta xây dựng có độ chính xác cao (trên 97%) và không gặp tình trạng overfitting hay underfitting.

4.6 So sánh với MobileNet

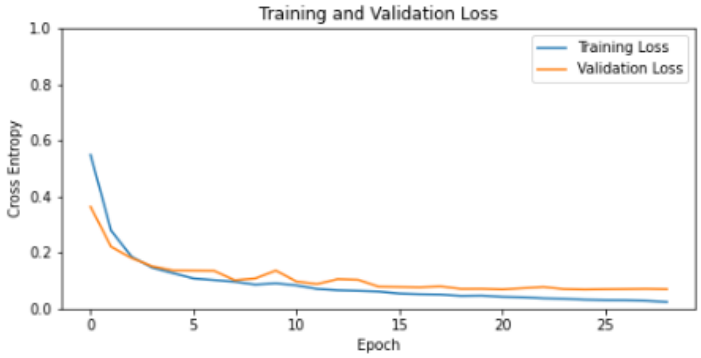
Về kết quả huấn luyện:

- Đồ thị accuracy và loss của MobileNet:





- Đồ thị accuracy và loss của Lenet:



Về độ chính xác trên tập Test

- MobileNet:

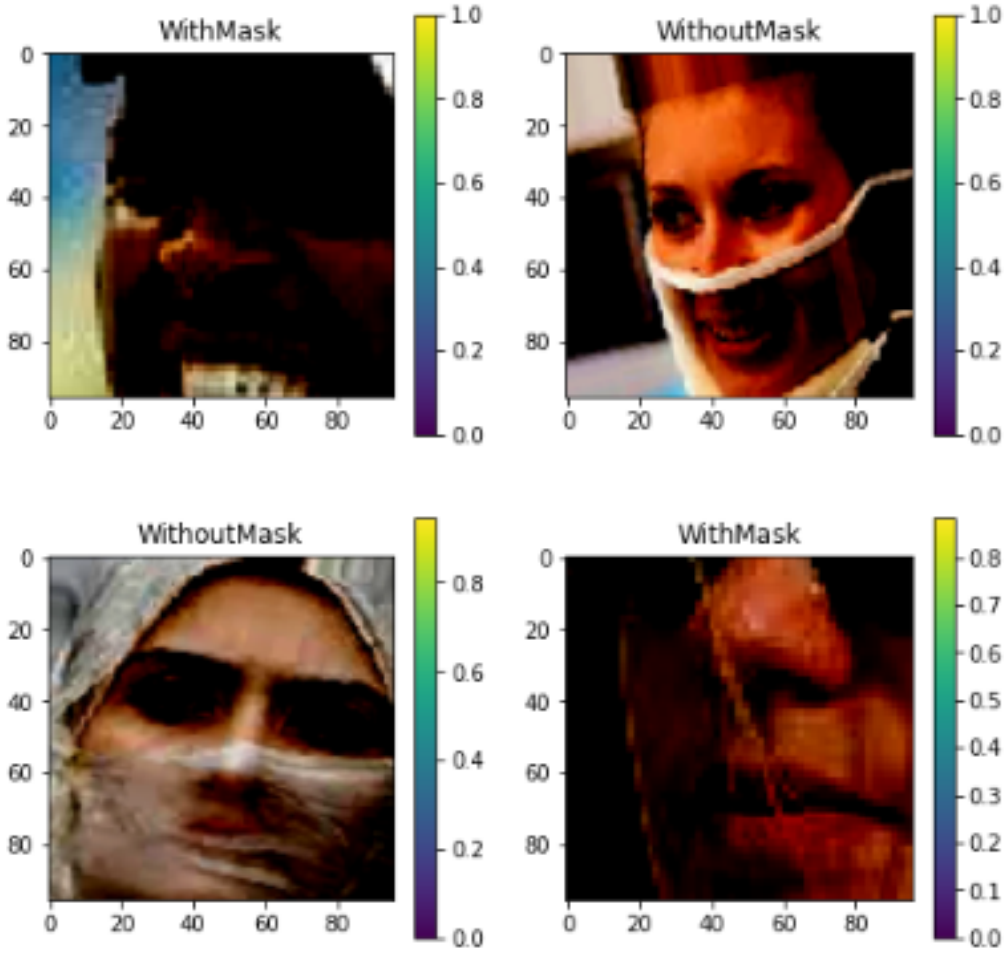
```
74/74 [==============================] - 14s 191ms/step - loss: 0.0146 - accuracy: 0.9970
Model accuracy: 99.70%
```

- Lenet:

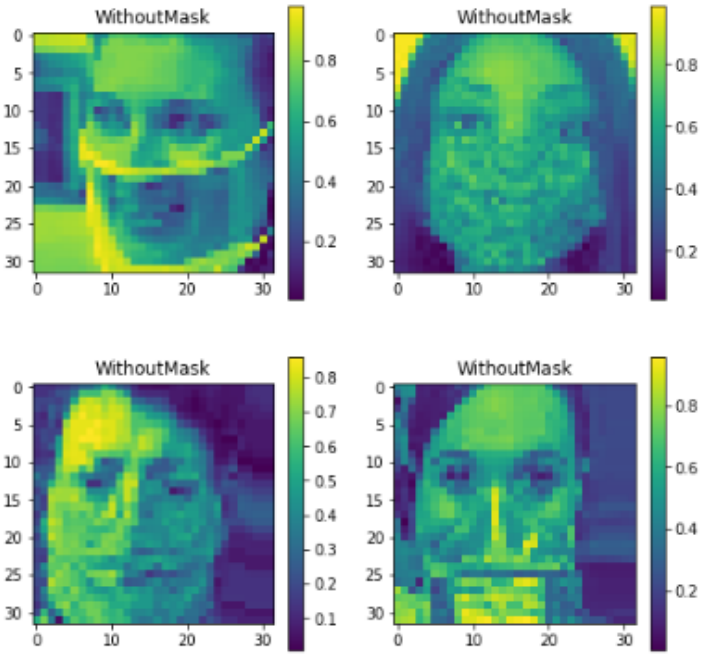
```
74/74 [==============================] - 1s 12ms/step - loss: 0.0819 - accuracy: 0.9729
Model accuracy: 97.29%
```

1 số hình ảnh nhận diện sai trong tập Test của 2 mô hình

- MobileNet:



- Lenet:



4.7 Đánh giá

Nhìn chung, mô hình MobileNetV2 vượt trội hơi Lenet-5 về mọi mặt.

- Mô hình MobileNetV2 đạt độ chính xác lên đến hơn 99% chỉ sau 10 epoch, độ chính xác của Lenet-5 chỉ đạt 97%
- Những hình ảnh MobileNetV2 nhận diện sai nguyên nhân chủ yếu do ảnh quá gần mặt hoặc khẩu trang trong suốt, mô hình này chỉ nhận diện sai 7 ảnh trên 2359 ảnh của tập Test.
- So với mạng MobileNet ra đời sau, mạng Lenet còn khá đơn giản nên dẫn đến một số sai sót trong việc xác định; ngoài ra một số ảnh khi bị thu nhỏ qua giai đoạn tiền xử lý dữ liệu của mạng Lenet bị vỡ, khiến việc xác định trở nên khó khăn.

5 Kết luận

5.1 Kết quả đạt được

Sau khi hoàn thiện bài toán 2: Nhận diện và phân loại hình ảnh đầu vào có phải là người đang đeo khẩu trang hay không, nhóm đã đạt được một số điểm:

- Nắm được cấu trúc mạng CNN; các đặc điểm của một bài toán nhận dạng nói chung và nhận dạng khẩu trang nói riêng
- Tìm hiểu được một số phương pháp nhận dạng ,nắm được ưu nhược điểm của từng phương pháp

- Sử dụng các thư viện cho việc phân loại hình ảnh
- Demo và Test thử thành công trên các bộ dữ liệu mẫu

Ưu điểm của Lenet: thuật toán đơn giản, tỉ lệ thành công khá cao.

Tuy nhiên, giải thuật còn tồn tại một số điểm hạn chế chưa giải quyết được như: Mô hình còn khá đơn giản với chỉ với 7 lớp, ảnh có thể bị vỡ khi kích thước đầu vào lớn.

5.2 Hướng phát triển

Tiếp theo, có thể áp dụng thêm bài toán 1: Trích xuất được vùng có khuôn mặt xuất hiện trong khung hình (đầu vào: hình ảnh, đầu ra: hình ảnh khuôn mặt chính diện). Từ đó có thể quét được khung hình rộng hơn, tiết kiệm thời gian.

Sau đó, có thể áp dụng thêm bài toán 3: Kết nối bài toán 1 và bài toán 2, thực hiện việc trích xuất và nhận diện khuôn mặt trong điều kiện thời gian thực thông qua camera/webcam.

Cuối cùng, phát triển thành ứng dụng cụ thể như: giám sát người tham gia xe buýt; giám sát người dân nơi công cộng; giám sát người làm việc tại bệnh viện, nhà máy thực phẩm, ...

6 Thành Viên và Khối lượng công việc

Họ và Tên sinh viên	**Mã số sinh viên**	**Công việc**	**Mức độ hoàn thành**
Phạm Hoàng Đức Huy	2011286	Viết báo cáo. Code, giải thích code	100%
Nguyễn Huỳnh Anh Duy	2011007	Viết báo cáo. Lí thuyết về Mạng Nơ-ron và CNN	100%
Hoàng Ngọc Trí	2014845	Viết báo cáo. Lí thuyết về mô hình Lenet	100%
Hoàng Tiến Hải	2011152	Viết báo cáo. Lí thuyết về lan truyền ngược	100%

Tài liệu

[1] **Nghiên cứu về mạng neural tích chập và ứng dụng cho bài toán nhận dạng biển số xe** , http://lib.uet.vnu.edu.vn/bitstream/123456789/918/1/K20_KTPM_Le_Thi_Thu_Hang_luanvan.pdf?fbclid=IwAR24pOnsh5c1QX06wU7_0vdqZApWTx8HVTGTB6Ws3zVePWvT8t4zM-h1POk

[2] **Multi-layer Perceptron và Backpropagation**, https://machinelearningcoban.com/2017/02/24/mlp/#-backpropagation

[3] **Convolutional neural networks: an overview and application in radiology**, https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[4] **LeNet-5 CNN with Keras - 99.48%**, https://www.kaggle.com/code/curiousprogrammer/lenet-5-cnn-with-keras-99-48/notebook